VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

NÁVRH DETEKTORU DOPRAVNÍCH ZNAČEK POMOCÍ METOD ZPRACOVÁNÍ OBRAZU
DESIGN OF TRAFFIC SIGN DETECTOR USING IMAGE PROCESSING METHODS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Josef Šmíd**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Stanislav Věchet, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Josef Šmíd**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**doc. Ing. Stanislav Věchet, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Návrh detektoru dopravních značek pomocí metod zpracování obrazu

Stručná charakteristika problematiky úkolu:

Mobilní roboty určeny pro pohyb ve venkovním prostředí využívají krom standardních senzorů jako jsou lidary nebo ultrazvukové dálkoměry také kamery. Zpracování obrazových dat z kamer, umístěných na robotu, v některých případech slouží jako jeden z hlavních senzorických vstupů do navigačních metod. Cílem této práce je zpracovat metodologii detekce dopravních značek s možností využít tyto informace pro navigaci mobilních robotů.

Cíle diplomové práce:

1. Proveďte rešerši aktuálně používaných metod detekce dopravních značek.
2. Sestavte testovací množinu dat z dostupných dopravních značek.
3. Proveďte srovnávací experimenty vhodných metod nad testovací množinou dat.
4. Navrhněte softwarový modul s definovaným API pro externí použití.
5. Vytvořený detektor otestujte a zhodnoťte výsledky provedených experimentů.

Seznam doporučené literatury:

SOLEM, J. E., Programming Computer Vision with Python: Tools And Algorithms For Analyzing Images, 1st edition, 2012.

LIGHT, R. A., Mosquitto: server and client implementation of the MQTT protocol, The Journal of Open Source Software, vol. 2, no. 13, May 2017, DOI: 10.21105/joss.00265.

BALÁTĚ, J.: Technické prostředky automatického řízení. Praha, SNTL 1986.

ROS.org. ROS.org | Powering the world's robots. [online]. 2.11.2016 [cit. 2016-11-02]. Dostupné z: http://www.ros.org/

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato diplomová práce se zabývá návrhem detektoru dopravních značek pomocí metod zpracování obrazu. K tomu je využita knihovna OpenCV pro práci s obrazem v programovacím jazyce Python. První část se zabývá rešerší o používaných metodách. V další části jsou tyto metody testovány na nasbíraných snímcích dopravních značek v běžném denním provozu při různém nasvícení. Z výsledků testů byly navrženy optimální metody a jejich nastavení, které se následně opět ověřily na videozáznamech jízdy vozidlem. Tím se zároveň zjistilo, za jakých podmínek jsou schopny fungovat v real-time systémech. Na závěr byl ze sledování průběhů detekce navržen optimalizační algoritmus pro kompenzaci chyb v detekci.

Klíčová slova

Autonomní vozidla, zpracování obrazu, detekce dopravních značek, OpenCV, Python

Abstract

This master thesis deals with the design of a traffic sign detector using the image processing methods. The OpenCV library for working with images in programming language Python is used for this. The first part reports on the using methods. In the next part, these methods were tested on images of traffic signs taken in traffic in different lighting conditions. The results of these tests led to the design of optimal methods and their settings, which were re-verified by verifying on video of driving in traffic. This also revealed the conditions under which they can operate in real-time systems. Finally, an optimization algorithm for compensation of detection errors was proposed from the monitoring of detection waveforms.

Keywords

Autonomous vehicles, image processing, traffic sign detection, OpenCV, Python

Bibliografická citace

ŠMÍD, Josef. *Návrh detektoru dopravních značek pomocí metod zpracování obrazu* [online]. Brno, 2020 [cit. 2020-06-03]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/124881. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Stanislav Věchet.

Prohlášení

Prohlašuji, že jsem diplomovou práci s názvem Návrh detektoru dopravních značek pomocí metod zpracování obrazu vypracoval samostatně pod vedením doc. Ing. Stanislava Věcheta, Ph.D., s použitím odborné literatury a pramenů uvedených v seznamu použitých zdrojů.

V Brně, 26. června 2020

……

Bc. Josef Šmíd

Poděkování

Tímto bych rád poděkoval svému vedoucímu diplomové práce doc. Ing. Stanislavu Věchetovi, Ph.D. za jeho ochotu a vstřícný přístup při řešení dílčích částí práce.

Obsah

1. Úvod 11
2. Cíle práce 11
3. Shrnutí používaných metod 12
   3.1 Metody strojového učení 12
   3.2 Metody zpracování obrazu 12
   3.3 Preprocessing 13
   3.4 Detekce 14
      3.4.1 Detekce segmentací barev 14
      3.4.2 Detekce hledáním tvarů 14
   3.5 Rozpoznávání 15
4. Analýza problému 16
5. Sběr testovacích vzorků 17
6. Rozpoznávání s detekcí 18
   6.1 MatchTemplate 18
      6.1.1 Test metody 19
      6.1.2 Výsledky 20
7. Detekce 22
   7.1 Detekce rozpoznáváním okrajů 22
      7.1.1 Test metody 23
      7.1.2 Výsledky metody 25
   7.2 Detekce prahováním obrazu 33
      7.2.1 Test metody 34
      7.2.2 Výsledky 36
   7.3 Detekce segmentací barev 40
      7.3.1 Test metody 41
      7.3.2 Výsledky 42
8. Rozpoznávání tvarů 46
   8.1 Rozpoznávání pomocí korelace 46
9. Rozpoznávání 49
   9.1 MatchTemplate 49
      9.1.1 Test metody 49
      9.1.2 Výsledky 51
10. Optimalizace 53

10.1 Ověření metody hledání okrajů 53
10.2 Ověření metody prahování 55
10.3 Ověření metody separování barev 56
11. Koncepční návrh 57
11.1 Vytvoření API 57
11.2 Návrh algoritmu pro eliminaci falešných pozitiv 59
12. Závěr 61
POUŽITÁ LITERATURA 63
SEZNAM OBRÁZKŮ 66
SEZNAM TABULEK 68
SEZNAM ZKRATEK 69
ELEKTRONICKÉ PŘÍLOHY 70

1. Úvod

Dopravní značky jsou nedílnou součástí silničního provozu. Upravují jízdní předpisy a dávají důležité informace například o přizpůsobení rychlosti nebo stavu vozovky. Díky tomu lze předcházet dopravním nehodám, ovšem jen pokud jsou tyto předpisy dodržovány. Systémy pro detekci dopravních značek se tedy využívají pro upozornění řidiče na určitou značku, kterou může přehlédnout. Ve spojení s dalšími systémy může navíc vozidlo zareagovat samo a například zpomalit nebo úplně zastavit. V případě plně autonomních vozidel musí tyto systémy dodržovat stejná pravidla silničního provozu, jako ostatní řidiči, a proto je nutné, aby byly schopné všechny dopravní značky správně a včas detekovat a vyhodnotit tak situaci.

K tomu slouží analýza obrazu v reálném čase. Jedná se tedy o tzv. real-time systém. Detekci lze provést mnoha způsoby zpracováním daného obrazu. Mohou to být například metody strojového učení, které se „učí" samy detekovat a rozpoznávat dopravní značky na základě několika vzorků.

Tato práce se však zabývá pouze jednoduchými metodami, kde se k detekci využívají typické vlastnosti dopravních značek, které je odlišují od okolí. Je k tomu využita knihovna OpenCV v programovacím jazyce Python. Ta obsahuje mnoho funkcí, jež lze k detekci i následnému přesnému rozpoznání využít. Experimentálně je zde zjišťováno jejich nejoptimálnější nastavení na nasbíraných vzorcích vybraných dopravních značek. Toto nastavení by poté mělo fungovat i na ostatní dopravní značky, což se ověří při použití v běžném provozu.

2. Cíle práce

1. Provést rešerši aktuálně používaných metod detekce dopravních značek.
2. Sestavit testovací množiny dat z dostupných dopravních značek.
3. Provést srovnávací experimenty vhodných metod nad testovací množinou dat.
4. Navrhnout softwarový modul s definovaným API pro externí použití.
5. Otestovat vytvořený detektor a zhodnotit výsledky provedených experimentů.

3. Shrnutí používaných metod

Jak již bylo zmíněno, obecně existuje mnoho způsobů pro rozpoznávání dopravních značek v obraze. Je to, jak strojové učení s využitím například neuronových sítí, tak různé metody zpracování obrazu, nebo jejich kombinace.

3.1 Metody strojového učení

Právě strojové učení je v dnešní době velmi často využívaná metoda. Zabývá se jí mnoho studií, ovšem kvůli velkému počtu testovacích vzorků a složitosti nastavení se tyto metody omezují pouze na určitý počet značek nebo na jejich konkrétní typ [1] [2]. Tím se rozumí například dopravní značky omezující rychlost, které jsou všechny stejné, mění se pouze jejich vnitřní číselná hodnota.

Protože jsou v dnešní době tyto systémy již běžně využívané ve vozidlech jako informace pro řidiče, je zde velká snaha o maximální spolehlivost v určování všech dopravních značek. Výběrem značek, které má být systém schopen rozpoznat, lze dosáhnout vysoké spolehlivosti, a to až 98 % [2]. Je však nutné podotknout, že se tyto hodnoty vztahují na již detekované značky. V procentech spolehlivosti není tedy zahrnutý proces detekce, který má také vliv na celkové množství rozpoznaných značek.

K tomu je možné využít jinou metodu strojového učení, a to například Haar Cascade [3], která na základě vstupních vzorků vytvoří modely pro detekci v dalších obrazech. Ta ovšem vyžaduje opět velké množství dat a výpočetního času pro vytvoření těchto modelů detekce. Většina systémů proto využívá pouze základní metody zpracování obrazu, například hledáním okrajů nebo separováním barev jednotlivých značek [4].

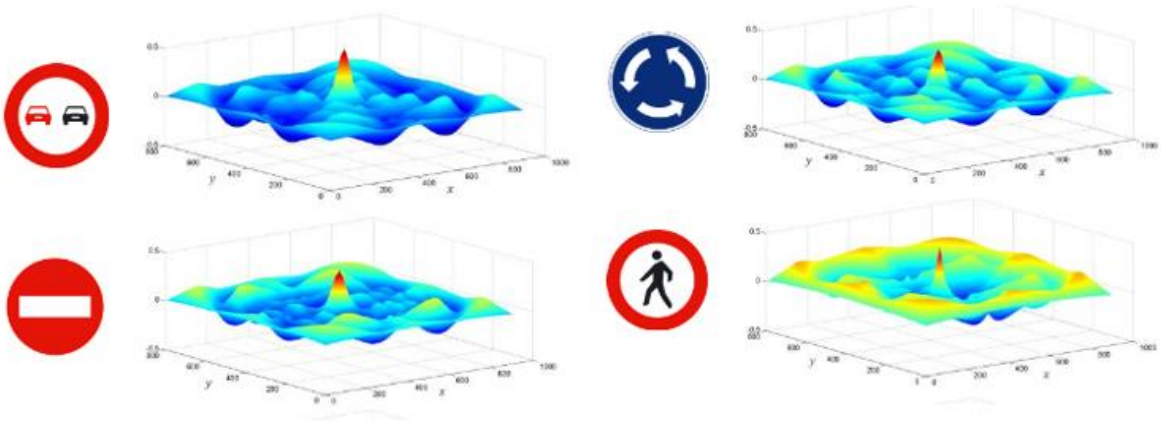
Nevýhodou metod strojového učení je tedy především to, že vyžadují velké množství vzorků pro učení. Pro každou další značku je pak nutné nasbírat další vzorky a systém se ji musí znova naučit. Tato síť pak musí být navíc schopná rozpoznávání v reálném čase, pokud by měl být systém použitý v běžném provozu.

3.2 Metody zpracování obrazu

K usnadnění postupů „učení" lze tedy využít statistické metody, které na základě porovnání pixelů vyhodnotí procento shody mezi šablonou a detekovanou značkou. Podle procenta této shody lze pak vyhodnocovat, o jakou značku se jedná, a to bez nutnosti sbírání vzorků, ale pouze s využitím její šablony.

K detekci se opět využívají již zmíněné metody hledáním okrajů nebo separováním barev, čímž se určují oblasti detekce. Ty lze navíc zpřesnit metodami pro rozpoznávání detekovaného tvaru, který musí odpovídat tvaru dopravních značek [5].

Samotné určení detekované dopravní značky pak probíhá na základě jejího tvaru, který se přiřadí ke značce se stejným tvarem, nebo se navíc aplikuje metoda korelace a hledá se její nejvyšší bod (Obr. 3.1). Tím se ověří, zda se skutečně jedná o dopravní značku a případně o jakou [5] [6].



Obr. 3.1 Mapy korelačních hodnot detekovaných značek [5]

Velký vliv na spolehlivost výsledků má rozlišení obrazu, kde se snižujícím se počtem pixelů klesá i přesnost korelace. Tímto způsobem lze však, při správném nastavení dosáhnou přesnosti v průměru až 91 % [5]. Metoda korelace je tedy téměř srovnatelná s metodami „učení". Záleží zde však opět na množství analyzovaných dopravních značek.

Výhodou pro všechny tyto metody je to, že jsou již většinou určitým způsobem předprogramované ve zmíněné knihovně OpenCV. Ta obsahuje mnoho různých funkcí, jejichž správným nastavením vznikne požadovaný výsledek. Žádná tato funkce však není schopná přímé detekce nebo rozpoznání, ale vyžaduje nejdříve úpravu obrazu. Proto je nutné využít jednotlivé kroky od přípravy obrazu až po samotné rozpoznávání, na které lze tyto funkce aplikovat [7].

3.3 Preprocessing

Základem pro všechny způsoby je tedy nutná příprava obrazu, tzv. „preprocessing". Vstupuje zde totiž mnoho faktorů, které mohou mít negativní vliv na schopnost rozpoznávání. Řadí se zde ovlivnitelné faktory, jako je například rozlišení kamery nebo samotná metoda zpracovávání (online – real-time, nebo offline), ale i neovlivnitelné, a to především počasí a technický stav dané značky. Při návrhu se proto musí všechny tyto faktory uvažovat [7].

Jeden z hlavních problémů při řešení představuje počasí, které nelze nijak ovlivnit, ale přitom má výrazný vliv na vzhled dopravních značek při snímání kamerou. Déšť, sníh nebo mlha pak můžou úplně snížit jejich viditelnost. Podle intenzity osvětlení se navíc ještě mění odstíny jednotlivých barev. Dalším problémem je stav dané značky, který je ovlivněn například slunečním zářením, stárnutím nebo mechanickým poškozením způsobeným dopravou. To vše mění výslednou podobu dopravní značky a znemožňuje její detekci. Mezi ovlivnitelné faktory pak patří například velikost snímaného obrazu. Ta je dána rozlišením použité kamery. Při příliš malém rozlišení se výrazně snižuje schopnost rozpoznávání, naopak ale zase při příliš velkém rozlišení se prodlužuje doba výpočtu. V případě real-time systému je však nutné toto uvažovat a případně snížit rozlišení kamery na úkor schopnosti rozpoznávání [7].

3.4 Detekce

Detekci lze provádět pro urychlení výpočtů, ale zároveň i pro zlepšení jejich výsledků. Různými metodami na základě hledání tvarů, barev a dalších typických vlastností značek, se v obraze detekují pouze oblasti, ve kterých by se měla nacházet dopravní značka. K tomu lze opět využít metody zpracování obrazu v knihovně OpenCV nebo také strojové učení. S těmito oblastmi se pak při procesu rozpoznávání pracuje stejně jako s celým obrazem, ovšem s tou výhodou, že je již potlačené pozadí. Díky tomu vstupuje do výpočtu menší množství dat [7].

3.4.1 Detekce segmentací barev

Cílem je separování barev, které jsou typické pro dopravní značky. Určením jejich prahové hodnoty se oddělí od pozadí a vzniknou tak oblasti detekce. Využívají se k tomu různé grafické formáty založené jak na hodnotách jednotlivých barev (např. RGB – červená, zelená, modrá), tak na vlastnostech barvy (např. HSV – odstín, sytost, jas) [7] [8].

Při použití formátu HSV vykazuje sytost barvy výrazný vliv, pokud je snímek pořízen z různých zařízení. Vlivem osvětlení během dne i noci bývá barva obrazu zkreslená, což má za následek zhoršení kvality. To by měl částečně kompenzovat barevný formát RGB, kde na rozdíly v jednotlivých složkách nemá vliv venkovní osvětlením. Obecně je však segmentace barev označována za ne příliš spolehlivou, protože se vždy musí prahové hodnoty jednotlivých barev neustále upravovat [7].

Ke zvýšení schopnosti rozpoznávání barev lze použít metody učení na základě genetického algoritmu [9] nebo SVM algoritmu [10]. Tyto metody by měly být schopny dosáhnout lepších výsledků v porovnání se samotnou metodou segmentace barev [7].

3.4.2 Detekce hledáním tvarů

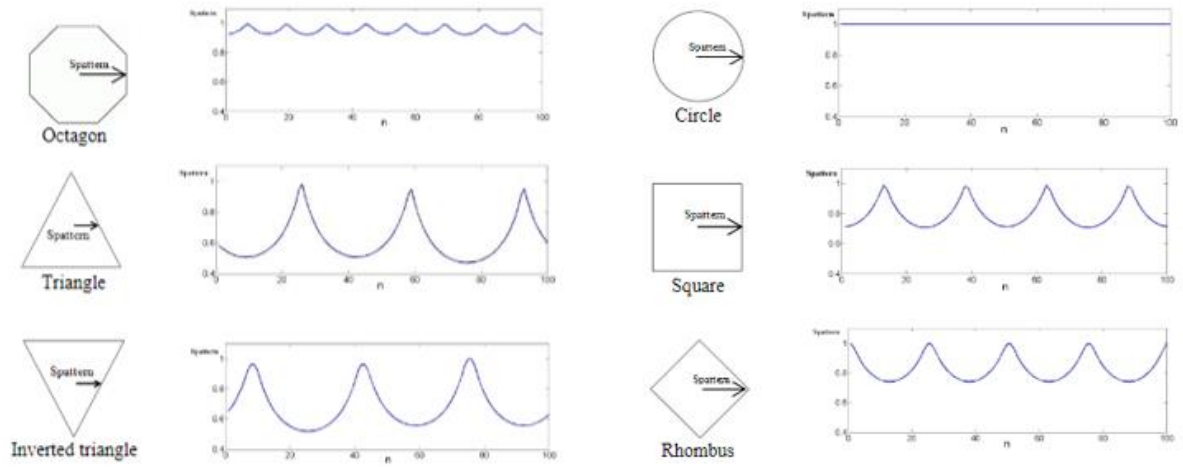
Vzhledem k tomu, že detekce pomocí barev často selhává z toho důvodu, že na přesnost barev jsou kladeny vysoké nároky, jichž není možné běžně dosáhnout, přistupuje se k metodám hledáním tvarů. Ty bývají většinou účinné na obrazech ve stupních šedi, což navíc snižuje i nároky na kameru. Tvar se pak zjišťuje z detekovaných hran jejich aproximací, porovnáním se šablonou nebo pomocí Houghovy transformace [7].

Tyto hrany lze určit zmíněnou metodou hledáním okrajů. Knihovna OpenCV k tomu má předepsanou funkci Canny Edge, která je založená na gradientní metodě. Stanovením dvou prahových hodnot se rozdělují oblasti, do kterých vzniklý gradient spadá. Podle toho se následně určí, zda se jedná o okraj, či nikoliv [12]. Vzhledem ke kontrastu dopravních značek s okolím je možné detekovat právě jejich okraje. Z nich pak lze například pomocí aproximace vzniklých kontur určit počet stran a tím rozpoznávat jednotlivé tvary [13].

Stejným způsobem lze také detekovat tvary s použitím metody prahování obrazu, tzv. „thresholding“. Při použití černobílého obrazu se nastaví prahová hodnota určitého stupně šedi. Ta následně rozdělí obraz na binární hodnoty. Díky tomu vzniknou opět tvary s konturami, ze kterých se pomocí aproximace rozpoznají [14]. V tomto případě lze částečně ovlivnit vliv počasí při použití adaptivního thresholdingu. Jak z názvu vyplývá,

tato metoda je schopná přizpůsobit se (adaptovat se) různým světelným podmínkám. Jeho použití je ovšem o něco náročnější z hlediska výpočetního času [15].

Určování tvarů porovnáváním šablon lze provést pomocí Hausdorffovy vzdálenosti mezi šablonou a snímaným obrazem. Každý tvar má svůj typický „podpis", který je porovnáván s podpisem detekovaného tvaru a vyhodnocují se rozdíly ve vzdálenostech mezi nimi (Obr. 3.2). Z důvodu velkých výpočetních nároků nelze tuto metodu použít v real-time aplikacích. Je ovšem vhodná po určitých úpravách pro mnoho jiných projektů, kde není nutný rychlý výpočet [7].



Obr. 3.2 „Podpisy" jednotlivých tvarů [5]

Poslední metody jsou, již zmíněná Houghova transformace a také metody strojového učení. Houghova transformace je schopná detekovat čáry a kruhy v obraze. Výhodou je, že není citlivá na nepřesnosti a šum. Strojové učení pak opět využívá například metodu SVM a neuronové sítě jako v předešlých příkladech [7].

3.5 Rozpoznávání

V případě metod, které využívají k rozpoznávání detekovaných oblastí šablonu, určuje shoda mezi nimi, o jakou dopravní značku se jedná. V případě, že je shoda vyšší než nastavená prahová hodnota, je tato značka detekována [6]. Nevýhodou v některých případech může být ovšem velikost šablony, natočení nebo částečné překrytí značky, což znemožňuje její rozpoznání [7].

Dalšími možnostmi jsou techniky strojového učení. Je to například SVM (Support Vector Machine) [10] nebo neuronové sítě [1] [7]. Tyto metody fungují na základě natrénovaných dat, kdy se na vstup dosadí obraz a na výstup požadovaný výsledek (dopravní značka). Při vložení jiného obrazu by s použitím těchto dat měla být na výstupu opět hledaná značka [4].

4. Analýza problému

Jak vyplývá z rešerše o aktuálně používaných metodách, existuje mnoho způsobů, jak detekovat a rozpoznávat dopravní značky v obraze. Protože je práce zaměřená především na jejich rozpoznávání pomocí standardních metod zpracování obrazu v knihovně OpenCV, lze pominout metody strojového učení.

Na základě poznatků z využívání korelační metody napříč detekovanou značkou (cross-correlation) lze využít tuto metodu pro detekci dopravní značky i její samotné rozpoznání. Při použití na celý obraz by se pak v bodě s nejvyšší korelační hodnotou měla nacházet. Nicméně většinou se celý proces rozpoznávání rozděluje do několika kroků. Nejdříve se detekují oblasti, ve kterých se s určitou pravděpodobností nachází dopravní značky. To se určuje na základě jejich typických vlastností, jako jsou barva nebo tvar. Díky kontrastním barvám vůči okolí lze tyto barvy separovat nebo detekovat jejich okraje. Další možností je prahování obrazu podle jednotlivých stupňů šedi. Tím vzniknou kontury, pomocí nichž lze analyzovat, o jaký tvar se jedná. Následně se určuje, co se v dané oblasti vyskytuje, pomocí rozpoznávacích metod.

Největším problémem jsou ovšem změny počasí nebo osvětlení. Barva na dopravní značce má ve dne jiný odstín než v noci, kdy je osvětlená pouze světlomety vozidla. Vlivem počasí pak může být zhoršená její viditelnost. S tím vším se musí počítat a jednotlivé metody ověřit pro různé denní situace. Je tak nutné nasbírat několik vzorků pro otestování vlivu změny osvětlení během dne i noci. Počasí, které ovlivňuje především celkovou viditelnost dopravních značek, není nutné v tomto případě uvažovat, protože tomu nelze nijak předcházet. Ověřením chování jednotlivých metod na několika vzorcích v různých denních situacích je tedy možné navrhnout výslednou aplikaci pro rozpoznávání dopravních značek.

5. Sběr testovacích vzorků

Protože je nutné ověřit funkčnost metod při různých denních situacích, jako je především slunečno, oblačno/zataženo, nebo také schopnost rozpoznávání ve tmě, musí se pro tyto situace nasbírat vzorky.

Pro autentičnost vzorků s reálnou situací vycházel jejich sběr pomocí videozáznamu jízdy vozidlem. Z těch se následně vystříhaly snímky dopravních značek. Tím se zároveň vytvořila i statistika četnosti každé z nich, která měla rozhodující charakter pro výběr testovacích vzorků.

Z těchto nasbíraných dat bylo vybráno pět typově rozdílných dopravních značek, které mají nejlépe otestovat chování všech metod na každou z nich. Jedná se především o značky s různou barvou a tvarem. Tyto typické vlastnosti mohou právě sloužit k detekci. V druhé řadě pak rozhodovala jejich četnost na silnicích, aby byly použité metody vyladěny především na ty nejběžnější.

Na základě těchto kritérií byly tedy vybrány následující značky, jako testovací vzorky (Obr. 5.1):

- Hlavní pozemní komunikace;
- Dej přednost v jízdě;
- Začátek/konec obce;
- Zákaz zastavení/stání;
- Přikázaný směr.

Obr. 5.1 Testovací vzorky dopravních značek [16]

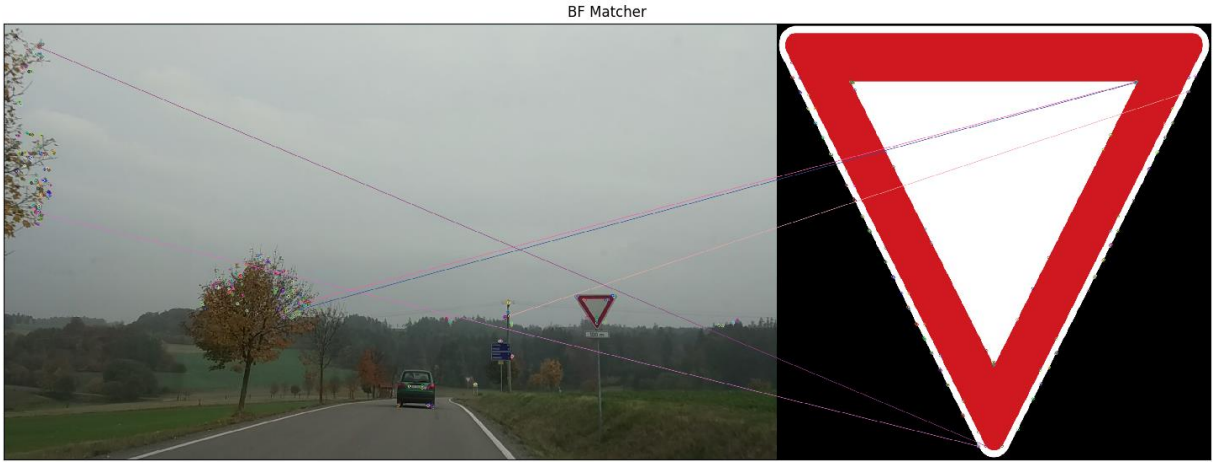
Hlavní pozemní komunikace a Dej přednost v jízdě byly zvoleny především z toho důvodu, že se jedná o nejběžnější a jedny z nejdůležitějších dopravních značek, protože upravují přednost v jízdě. Měly by být proto nejlépe rozpoznatelné. Značky začátku a konce obce, zákazu zastavení/stání a přikázaného směru se vybraly kvůli jejich různé barvě, aby bylo možné následně porovnat její vliv na detekci. Ve městech jsou navíc hodně rozšířené, mimo značky obce, a proto se zvolily právě tyhle. Všechny vybrané značky pokrývají hned čtyři různé tvary, a to čtverec, kosočtverec, kruh a trojúhelník.

6. Rozpoznávání s detekcí

Knihovna OpenCV obsahuje několik metod pro rozpoznávání věcí v obraze. Běžné pro tuto situaci jsou ty, které využívají šablonu pro porovnávání s testovaným obrázkem. Jsou to například metody BFMatch (Brutal force matching) nebo matchTemplate.

MatchTemplate hledá shodu se šablonou pomocí korelace, tedy hledáním vzájemného vztahu. To umožňuje při správném nastavení celkem spolehlivě detekovat šablonu v místě, kde má největší korelační koeficient. Je však nutné, aby šablona měla stejnou velikost a natočení jako v testovaném obrázku.

BFMatch oproti tomu hledá shodu se šablonou tzv. „hrubou silou" bez závislosti na velikosti a natočení. Shoda se hledá na základě podobných vlastností mezi obrázkem a šablonou. Touto metodou však nelze jednoznačně určit pozici hledané značky, protože v celém obraze vždy existuje mnoho podobných vlastností i mimo danou značku (Obr. 6.1). Tímto způsobem tak nelze rozpoznat rozdíly mezi více šablonami, a proto ji není vhodné použít pro tuto aplikaci.



Obr. 6.1 Detekované vzájemné body metodou BFMatch

6.1 MatchTemplate

Metoda umí pracovat jak s barevnými obrázky, tak i s černobílými. Černobílé obrázky zkracují čas výpočtu díky menšímu počtu hodnot, ale mohou snížit schopnost rozpoznání. Funkce má pouze tři parametry, tj. testovaný obrázek, šablonu a typ. Proto jediné, co je potřeba nastavit, je typ, který má dvě podoby, kovariační (základní) a korelační (normalizovanou). Korelační podoba vychází z kovariace, ale výsledný koeficient je zde převeden na podíl shody. Udává, jak moc si jsou dva snímky vzájemně podobné [17]. Díky tomu lze porovnávat více výpočtů mezi sebou.

Typy korelačních metod:

- CV_TM_CCOEFF_NORMED
- CV_TM_CCORR_NORMED
- CV_TM_SQDIFF_NORMED

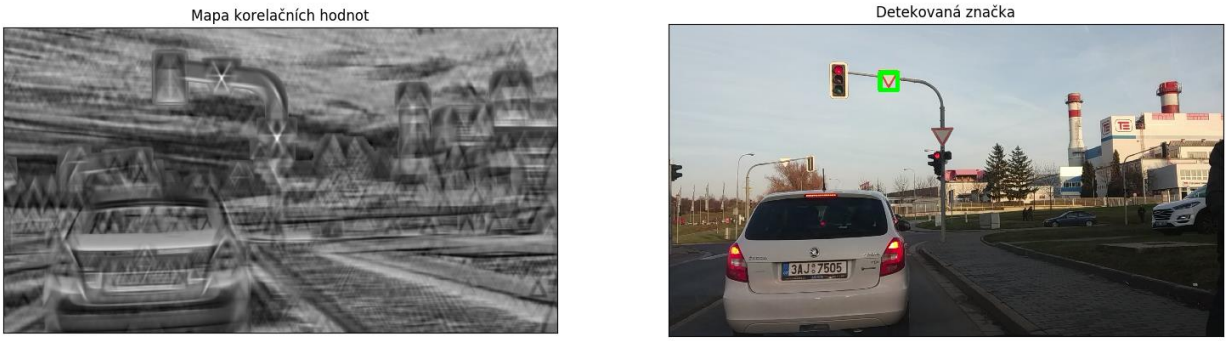
Výsledkem metody je pak mapa korelačních hodnot pro každý pixel. Z ní se následně určí pozice maximální hodnoty, která je zároveň i pozicí rozpoznané značky.

6.1.1 Test metody

Jak již bylo zmíněno, metoda matchTemplate používá pro rozpoznávání korelaci a potřebuje stejnou velikost a natočení originálu i šablony. To může být pro samotnou detekci problém, ale pro její otestování se kód upravil, aby porovnával i různé velikosti. Každá šablona byla tedy postupně testována v pěti různých velikostech od 60×60 px po 150×150 px na celkové velikosti obrazu 1920×1080 px. Tím se ovšem ale mnohonásobně zvýšil čas k detekci. Natočení v tomto případě není nutné řešit, protože ve všech běžných situacích jsou značky na silnici čelně k výhledu z vozidla. Pro ukázku byl vybrán následující obrázek (Obr. 6.2).

Obr. 6.2 Vzorový obrázek

Testovaly se všechny možnosti pro barevnou i černobílou variantu obrázku s použitím normalizovaných typů rozpoznávání, aby bylo možné vzájemně porovnávat výsledky. Na výsledných mapách korelačních hodnot značí bílá barva větší korelační shodu oproti černé (Obr. 6.3). Z každé takto vzniklé mapy se následně pomocí funkce „cv.minMaxLoc()“ určila maximální korelační hodnota a její pozice. Celkově nejvyšší výsledná korelace pak měla odpovídat hledané dopravní značce, v tomto případě „Dej přednost v jízdě“.

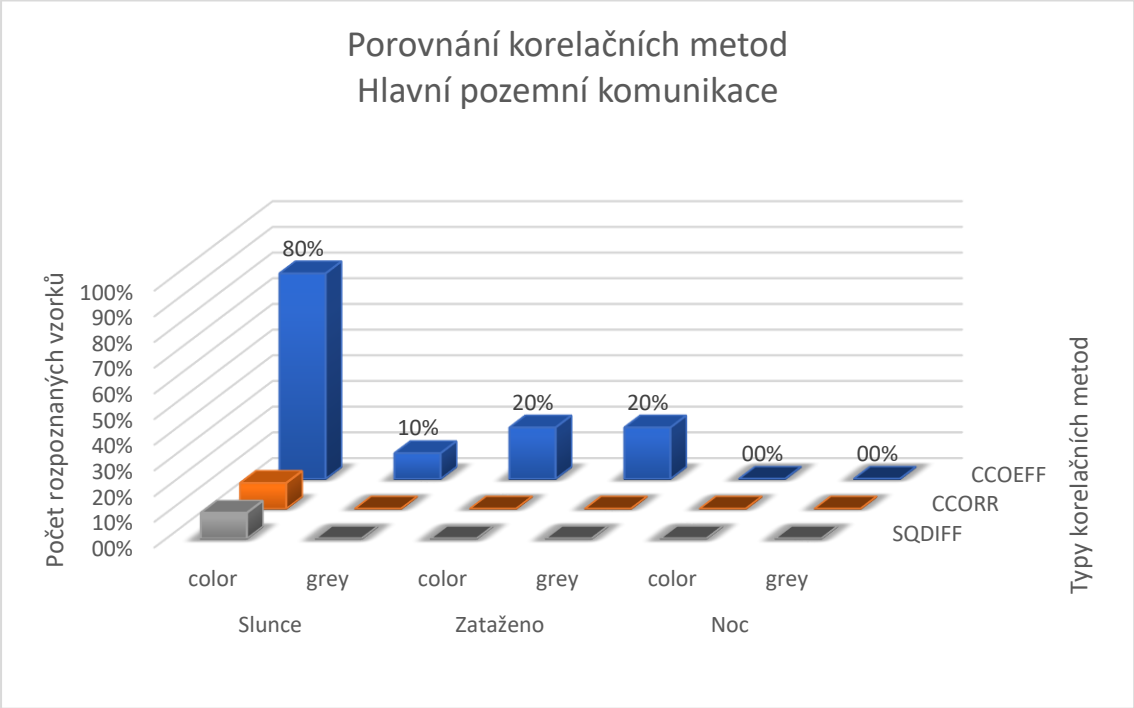


Obr. 6.3 Mapa korelačních hodnot (vlevo) a detekovaná značka (vpravo)

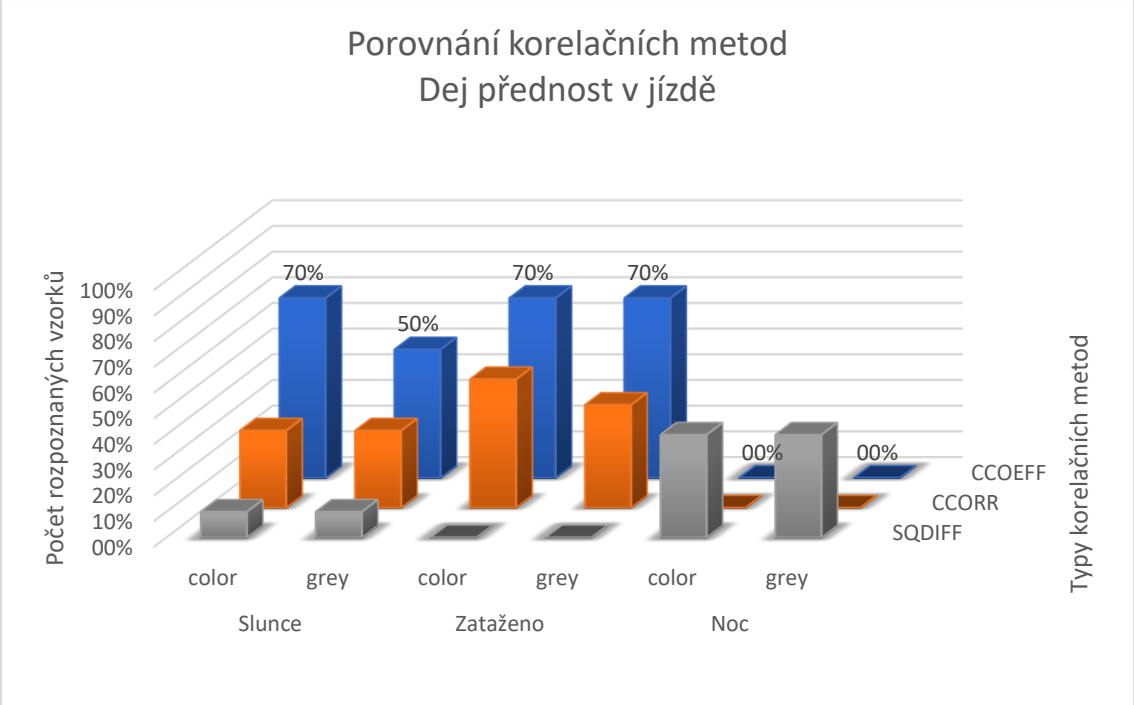
Protože bylo testováno použití metody tímto způsobem pouze pro ukázku spolehlivosti a časové náročnosti, zda má důvod se s ní vůbec zabývat, testovaly se pouze dva typy značek, a to „Hlavní pozemní komunikace“ a „Dej přednost v jízdě“.

6.1.2 Výsledky

Výsledky testování, které ukazují množství rozpoznaných značek, jsou shrnuty v grafech pro jednotlivé denní situace (Obr. 6.4–6.5).



Obr. 6.4 Porovnání korelačních metod – Hlavní pozemní komunikace



Obr. 6.5 Porovnání korelačních metod – Dej přednost v jízdě

Z testování vyplývá, že metoda obecně není vhodná, ani z hlediska časové výpočetní náročnosti, kdy se tato doba pohybovala přibližně od 1000 ms do 1200 ms pro barevný obrázek a od 550 ms do 800 ms pro černobílý, ani z hlediska funkčnosti. Lze ale částečně predikovat, že pro různé denní situace fungují různě spolehlivě všechny typy metody a že spolehlivost u černobílých obrázků je skutečné nižší, ale zároveň méně časově náročná.

Z těchto poznatků tedy vyplývá, že je nutné nejdříve detekovat samotné značky rychlejšími metodami pouze na základě jejich typických vlastností bez ohledu na jejich přesném vzhledu, teprve poté až použít některou z metod rozpoznávání pro přesné určení.

7. Detekce

Detekci lze provádět hledáním typických vlastností dopravních značek. Protože jsou značky většinou výraznější než okolí, jednou z metod detekce je hledání okrajů mezi těmito kontrastními plochami. Další možností je použití metody prahování, která rozdělí obrázek na binární hodnoty podle předem dané prahové hodnoty. Podobně lze také detekci provádět separováním barev, typických pro dopravní značky. Tím vzniká opět binarizovaný obrázek, v tomto případě je ale rozdělen podle definované barvy.

7.1 Detekce rozpoznáváním okrajů

Knihovna OpenCV obsahuje funkci „Canny Edge" pro detekci okrajů. Je možné použít jak barevnou, tak i černobílou verzi obrazu. Jediné, co je potřeba nastavit, je dolní a horní prahová hodnota pixelů, podle níž se vyhodnocují okraje. K tomu se používá gradient intenzity obrazu, který se získává použitím horizontálního a vertikálního Sobelova filtru (Obr. 7.1) [18].

Horizontální

1	2	1
0	0	0
-1	-2	-1

,

-1	0	1
-2	0	2
-1	0	1

Vertikální

Obr. 7.1 Příklad horizontálního a vertikálního Sobelova filtru o velikosti 3×3 [11]

Tím vzniknou dva samostatné gradienty intenzity G_x a G_y v jednotlivých osách. Jednoduchým výpočtem pomocí Pythagorovy věty (rovnice 1) se následně zjistí výsledný gradient G, případně i jeho úhel natočení (rovnice 2) [18].

$$G = \sqrt{G_x^2 + G_y^2} \tag{1}$$

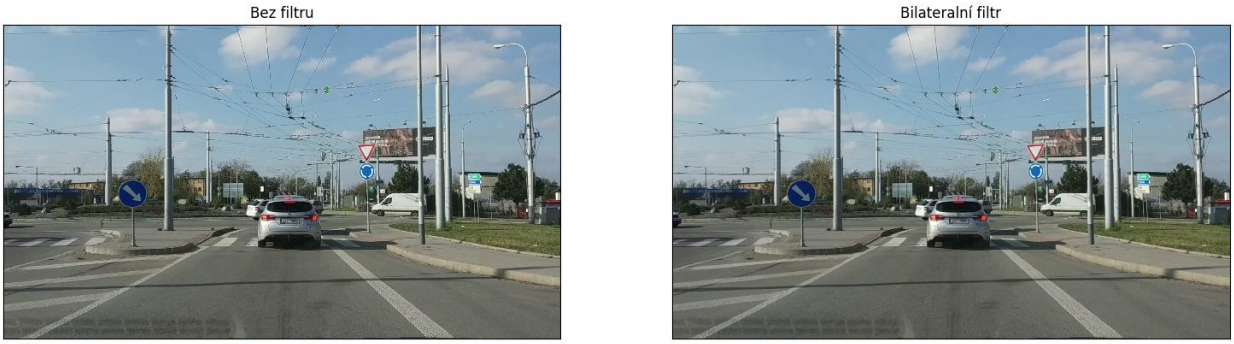
$$\varphi = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2}$$

Gradienty intenzity nad horní prahovou hodnotou jsou pak vyhodnoceny kladně, pod dolní zase záporně. Pokud se nachází mezi těmito hodnotami, záleží na vzájemném propojení mezi pixely, tedy zda souvisí s okraji, či nikoliv.

Protože je samotná detekce hran citlivá na šum, musí se obraz nejdříve vyfiltrovat některou z filtračních metod, jako jsou například Blur, MedianBlur, GaussianBlur a BilateralFilter, které obraz rozostří. Ty k tomu využívají konvoluci, tedy pronásobení obrazu konvolučním jádrem (kernelem), kde součet výsledných hodnot udává novou hodnotu pixelu. Toto jádro je dáno použitou metodou, nastavuje se pouze jeho velikost. U všech zmíněných metod vzniká celkově rozostřený obraz. Výjimkou je ovšem bilaterální filtr, který mimo rozostření navíc zachovává hrany. V nastavení se tedy s velikostí konvolučního jádra nastavuje také míra mixování barev a míra ovlivnění okolními pixely [18].

7.1.1 Test metody

Testovalo se především různé nastavení prahových hodnot. Pro každé toto nastavení se navíc testovaly i výše zmíněné filtrační metody. U každé z nich se zadala stejná velikost filtrační matice (3×3), aby se jejich vliv dal vzájemně porovnat. V rámci bilaterálního filtru se zbylé parametry volily zatím pouze intuitivně, na základě několika málo testů, tak aby byl obraz vyfiltrován co nejlépe. Pro porovnání a zjištění vlivu na detekci se navíc testovaly i obrázky bez filtrace (Obr. 7.2).

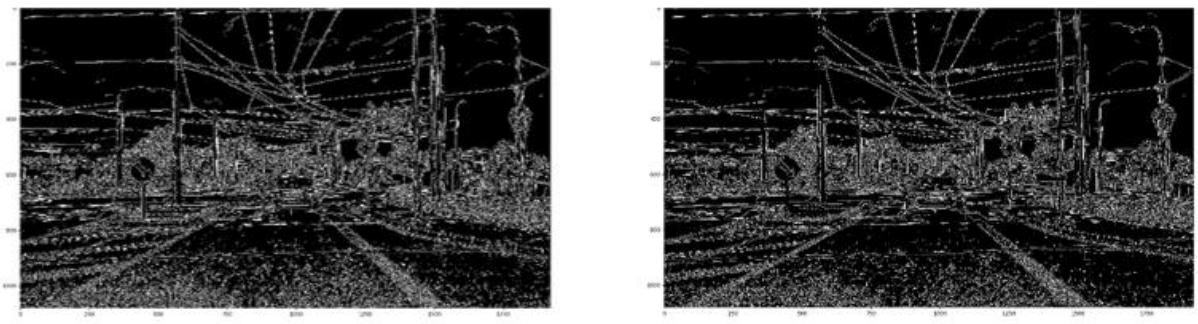


Obr. 7.2 Testovaný obrázek bez filtrace (vlevo) a s bilaterálním filtrem (vpravo)

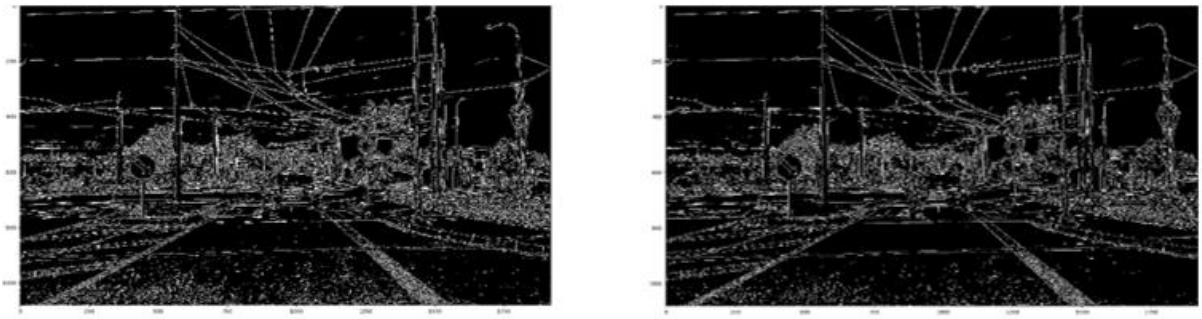
Na první pohled zde sice není rozdíl příliš viditelný, ale šum je skutečně odfiltrovaný. Je to patrné především na výsledcích. Poté se již aplikovala samotná detekce okrajů s různými parametry. Obrázek se zatím ponechal barevný, aby nedocházelo k nějakým ztrátám informace.

Prvním parametrem testování byla hodnota středu mezi požadovanými prahovými hodnotami. Ta se nastavovala tak, aby se při testování pokrylo, co možná nejvíce kombinací, které mělo smysl testovat. Pro zkoušku automatického rozpoznávání této hodnoty se za ni dosadily ještě medián, nebo průměr všech pixelů.

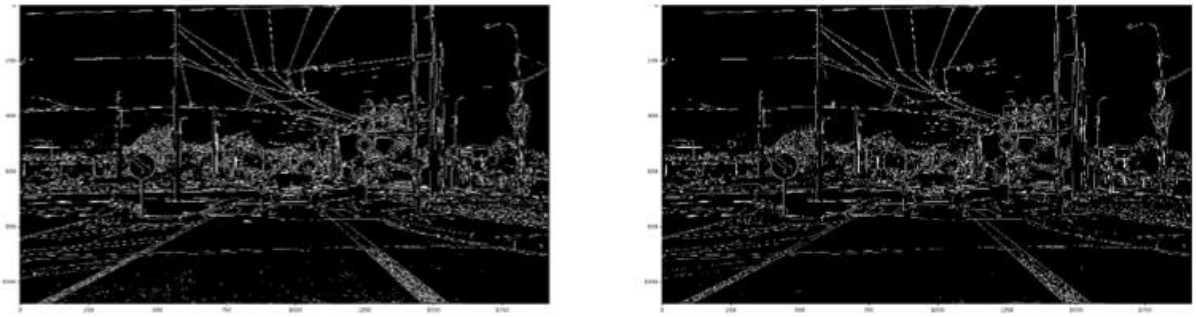
Dalším parametrem testování byl rozptyl, s jehož pomocí se dopočítaly obě prahové hodnoty. Celková velikost tohoto rozptylu se vždy měnila v závislosti na dané střední hodnotě, ze které se dopočítával. Jednalo se vždy o deset až padesát procent této hodnoty. Výsledné obrazy po detekci všech okrajů s různým nastavením střední prahové hodnoty vypadají pak následně (Obr. 7.3–7.5).



Obr. 7.3 Detekce okrajů bez filtrace (vlevo), s bilaterálním filtrem (vpravo)
(střední prahová hodnota – 30, rozptyl – 20 %)



Obr. 7.4 Detekce okrajů bez filtrace (vlevo), s bilaterálním filtrem (vpravo)
(střední prahová hodnota – 60, rozptyl – 20 %)



Obr. 7.5 Detekce okrajů bez filtrace (vlevo), s bilaterálním filtrem (vpravo)
(střední prahová hodnota – 120, rozptyl – 20 %)

Z takto vzniklého obrazu se musejí vytřídit oblasti detekce, tedy oblasti, ve kterých se s určitou pravděpodobností nachází dopravní značka. K tomu se využije funkce hledání kontur „cv.findcontours()“, kde se tento obraz použije jako vstupní parametr. Dále lze v této funkci nastavit ještě hierarchii vzniklých kontur, tj. vzájemná nadřazenost. Ta pak dokáže například rovnou eliminovat kontury, které se nachází uvnitř jiné (Opencv: cv.RETR_EXTERNAL), nebo je jen roztřídí (OpenCV: cv.RETR_TREE, cv.RETR_LIST,...) [19]. Pro zachování všech kontur se volila pouze metoda roztřídění. Nakonec lze nastavit ještě aproximaci (cv.APROX_SIMPLE). To znamená, že na výstupu budou pouze krajní body kontur, mezi nimiž je přímá vzdálenost. V opačném případě jsou bez aproximace (cv.APROX_NONE) na výstupu všechny body kontur. V tomto případě z hlediska funkčnosti na nastavení nezáleží, jde pouze o množství dat, se kterými se pracuje. Z toho důvodu se raději volilo použití aproximace.

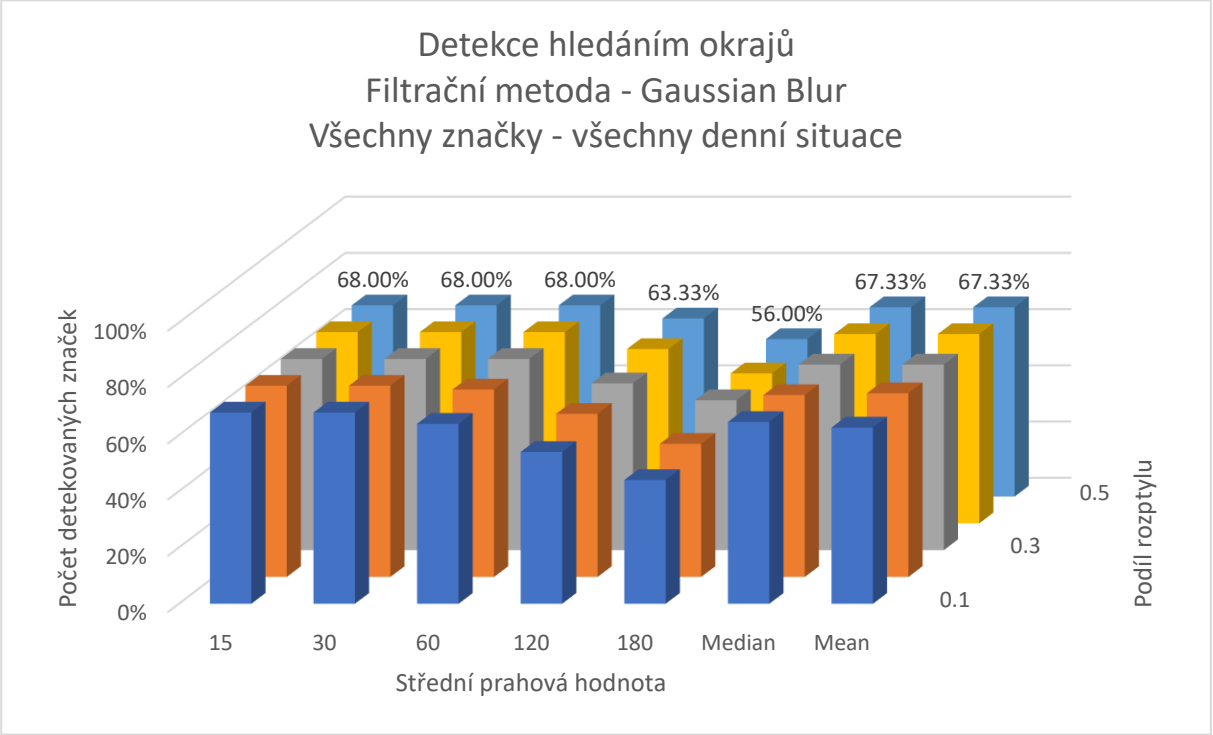
Z výsledného stavu po detekci kontur lze určit, že příliš malé kontury vzniklé zbylým šumem z okolí, nemůžou obsahovat dopravní značku. To platí i naopak pro příliš velké kontury překrývající většinu obrazu. Orámováním zbylých kontur pomocí funkce „cv.boundingRect()“, kde je vstupním parametrem pouze daná kontura, vzniknou oblasti detekce (Obr. 7.6–7.7).

Obr. 7.6 Detekované kontury (vlevo) a oblasti detekce (vpravo) bez filtru

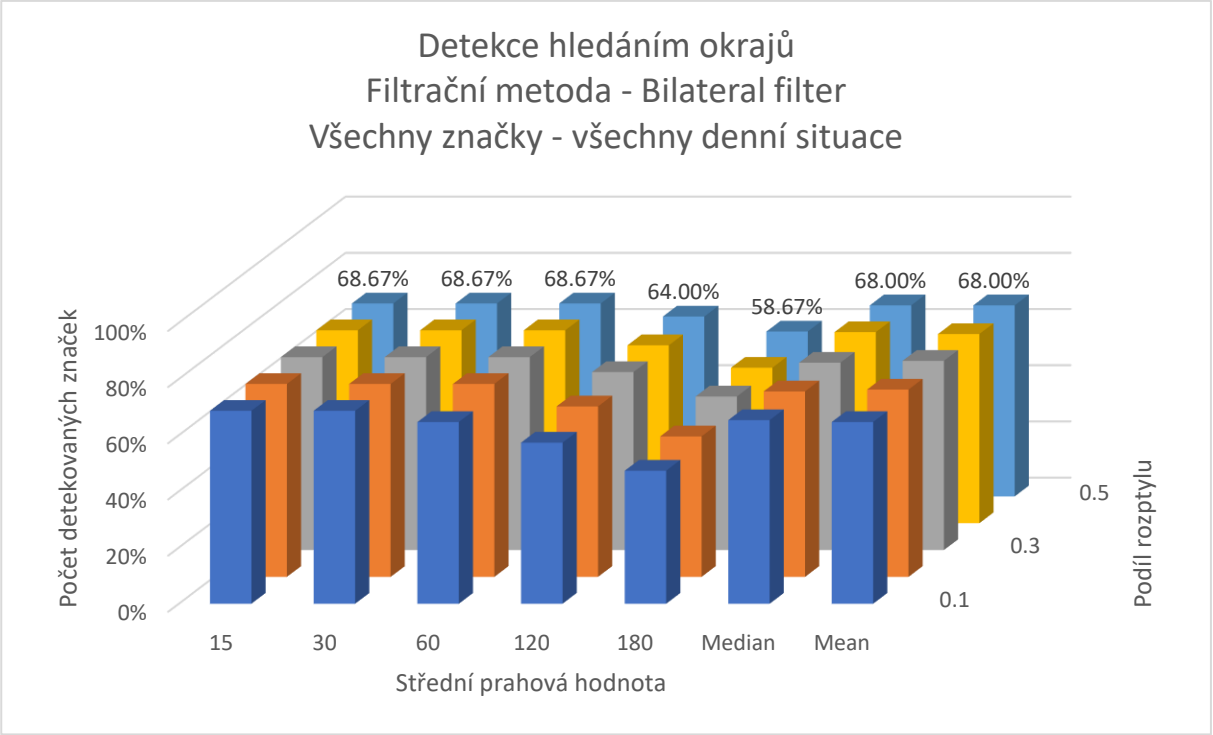
Obr. 7.7 Detekované kontury (vlevo) a oblasti detekce (vpravo) s bilatrálním filtrem

7.1.2 Výsledky metody

V následujících grafech je znázorněn průběh množství detekovaných značek pro různá nastavení ve všech denních situacích (Obr. 7.8–7.9). Jedná se však už jen o filtrační metody „GaussianBlur“ a „BilateralFilter“, protože dosahovaly největšího množství spolehlivých detekcí ze všech. Zbývající metody jsou znázorněny v elektronické příloze.



Obr. 7.8 Průběh detekcí hledáním okrajů pomocí Gaussova filtru



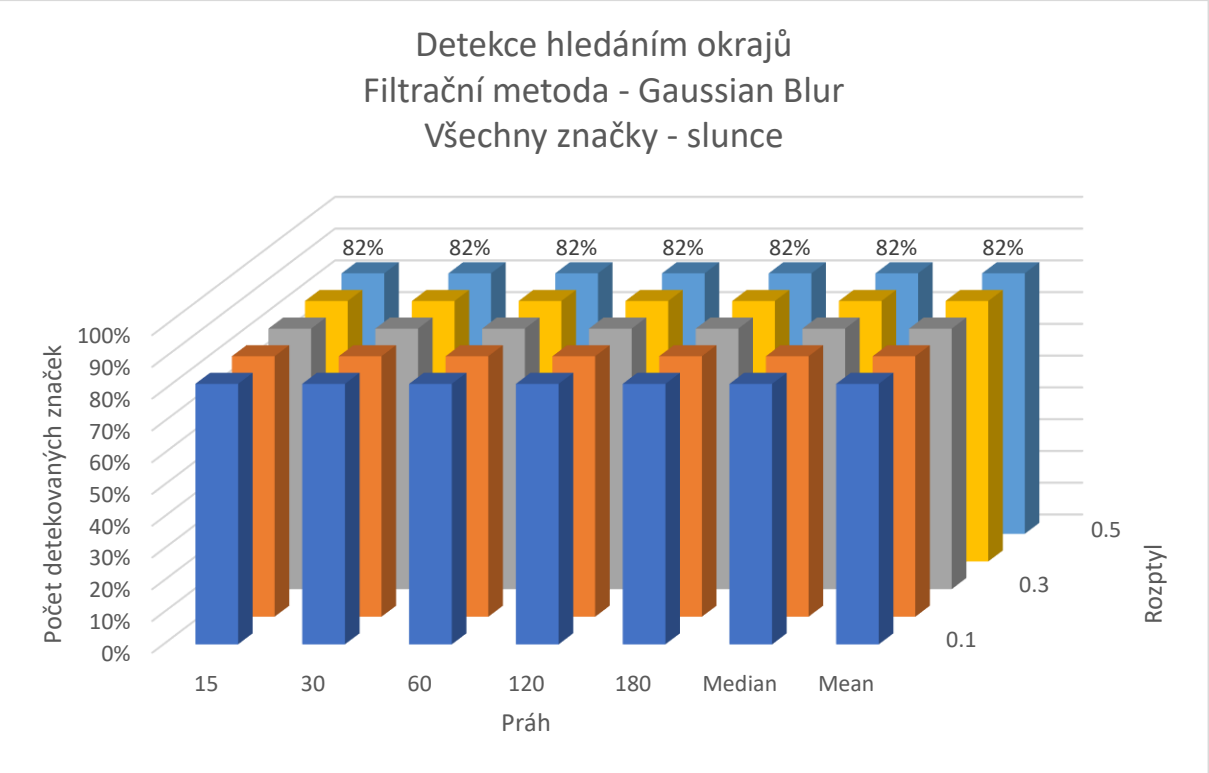
Obr. 7.9 Průběh detekcí hledáním okrajů pomocí bilateralniho filtru

Z daných grafů lze určit, že celková schopnost detekce je téměř 70 % u obou filtračních metod s minimálním rozdílem, proto je vhodné dále počítat s oběma variantami. Trochu lepších výsledků dosahovala detekce s použitím bilaterálního filtru, ovšem jeho výpočet trval při stejném nastavení vždy o něco déle než u Gaussova filtru. Celková doba detekce se s použitím filtračních metod pohybovala v rozmezí 40 ms až 180 ms, bez

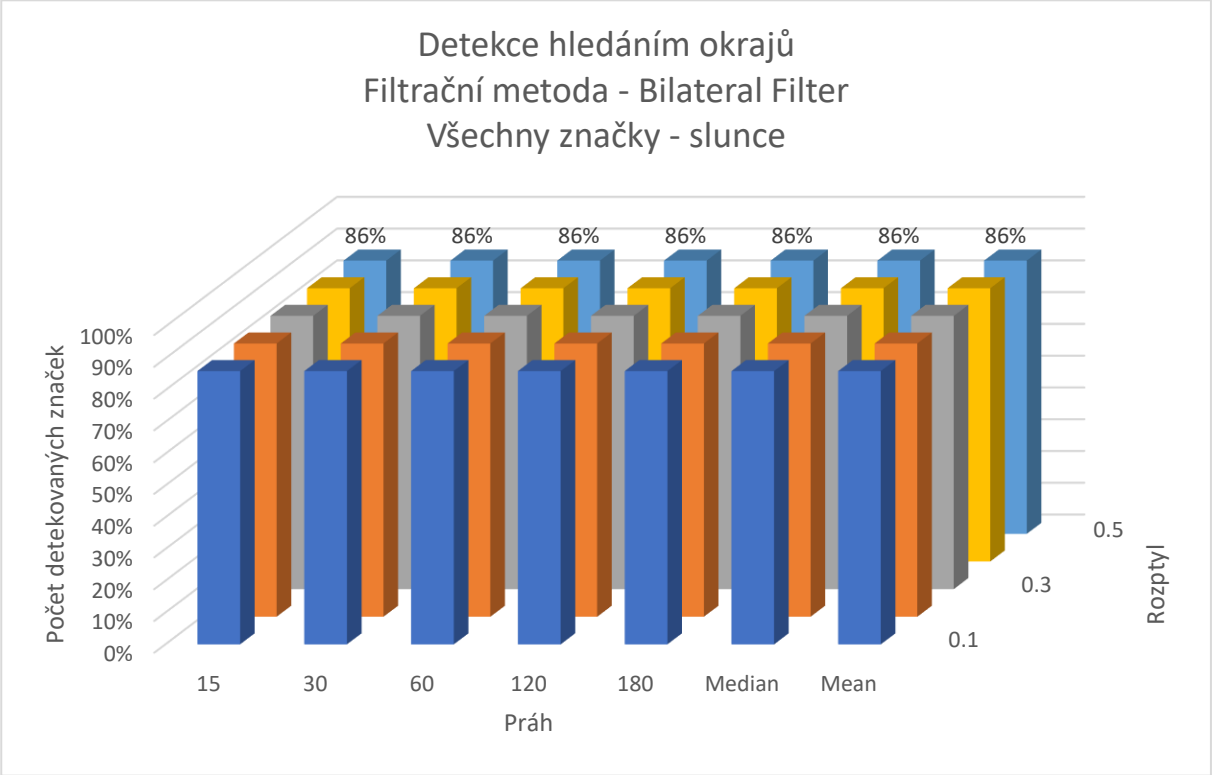
filtrace pak od 50 ms do 200 ms. Toto velké časové rozmezí je ovlivněno nastavením metody, kdy s nízkou střední prahovou hodnotou vzniká více detekovaných okrajů v obraze, s tím pak následně stoupá množství kontur a detekovaných oblastí. Tím se zvyšuje i doba výpočtu. Důležité je ovšem to, že použití filtrační metody vede celkově ke kratšímu výpočetnímu času, i když její provedení ho naopak zvyšuje. Je to dané opět tím, že odfiltrováním šumu vzniká menší množství detekovaných kontur.

Co se týče nastavení prahových hodnot jak u automatického určení střední prahové hodnoty, tak i u jejího pevného nastavení do hodnoty „60“, je spolehlivost algoritmu stejná. V ostatních případech klesá. Mění se ale množství detekovaných oblastí, kdy přibývá, nebo ubývá počet tzv. falešně pozitivních detekcí, tedy oblastí, které jsou detekované jako značky, ale ve skutečnosti jimi nejsou.

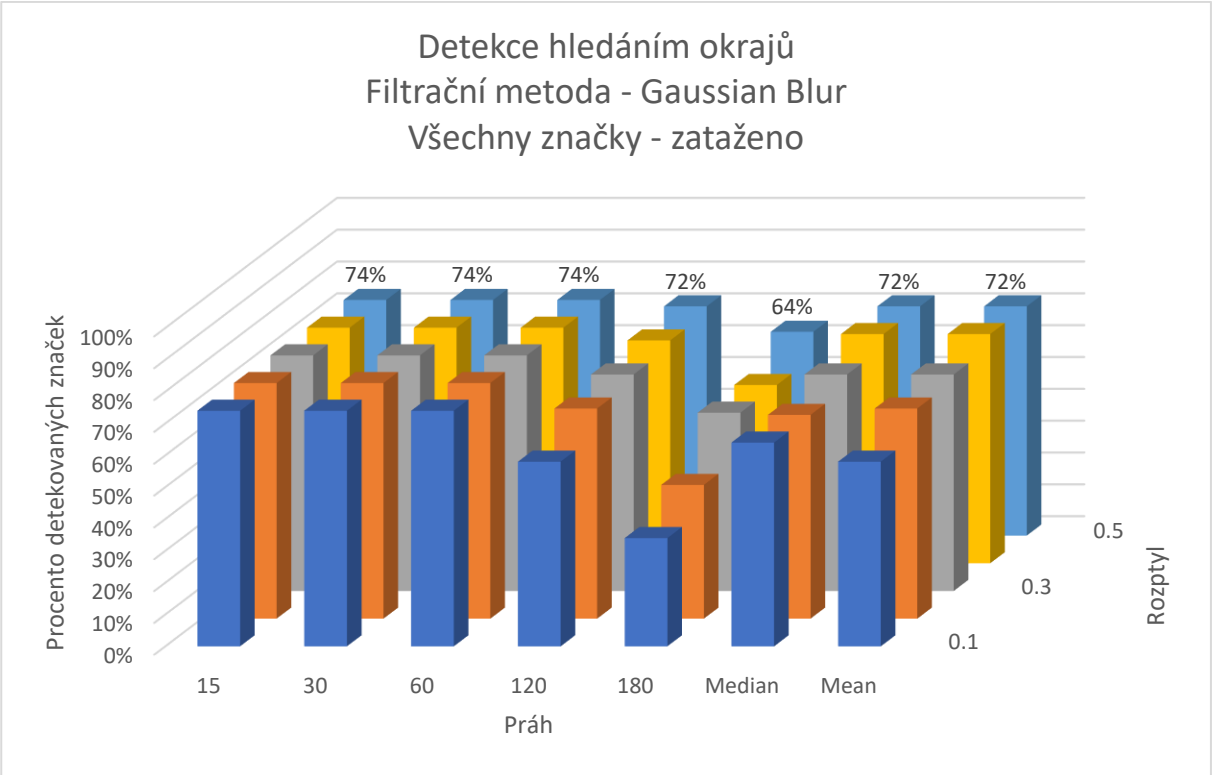
K vyvozování dalších a přesnějších závěrů je nutné znázornit, jak se tato metoda chová v jednotlivých denních situacích (Obr. 7.10–7.15). Jedná se opět o souhrn všech dopravních značek, ovšem teď jsou rozděleny zvlášť pro slunce, zataženo/oblačno a noc.



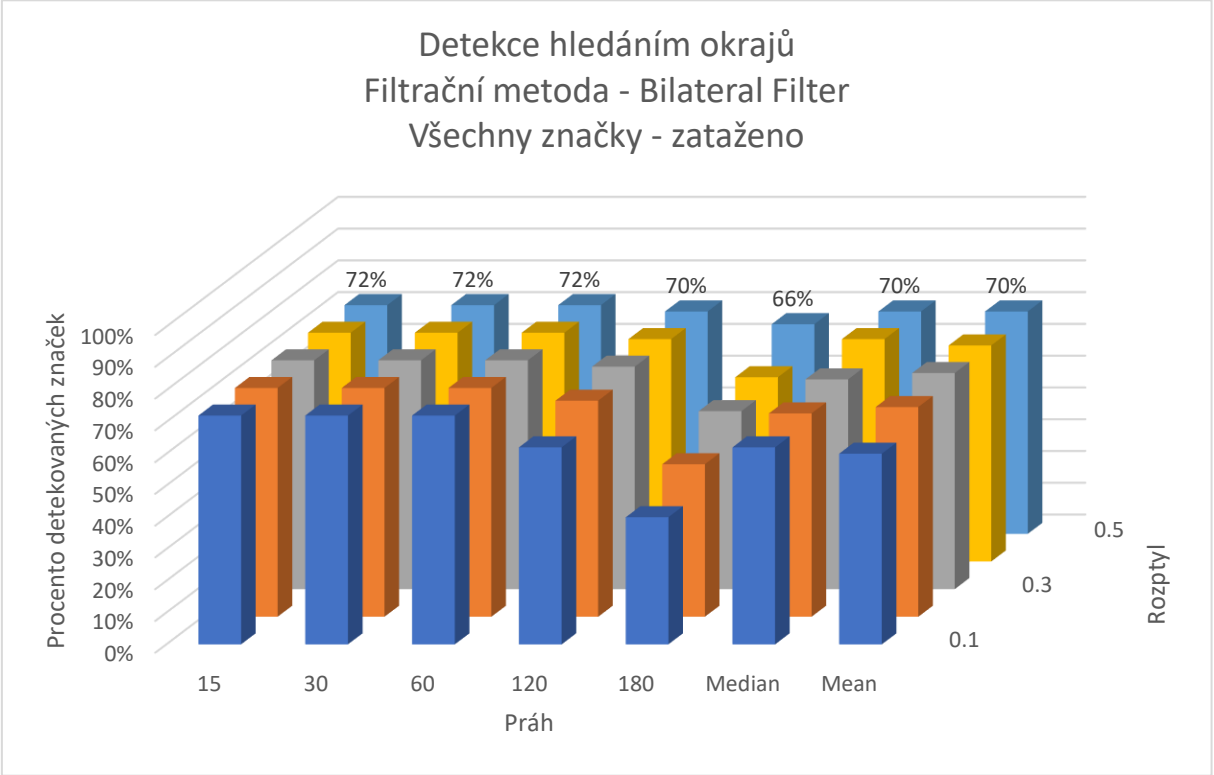
Obr. 7.10 Průběh detekcí hledáním okrajů pro slunečno pomocí Gaussova filtru



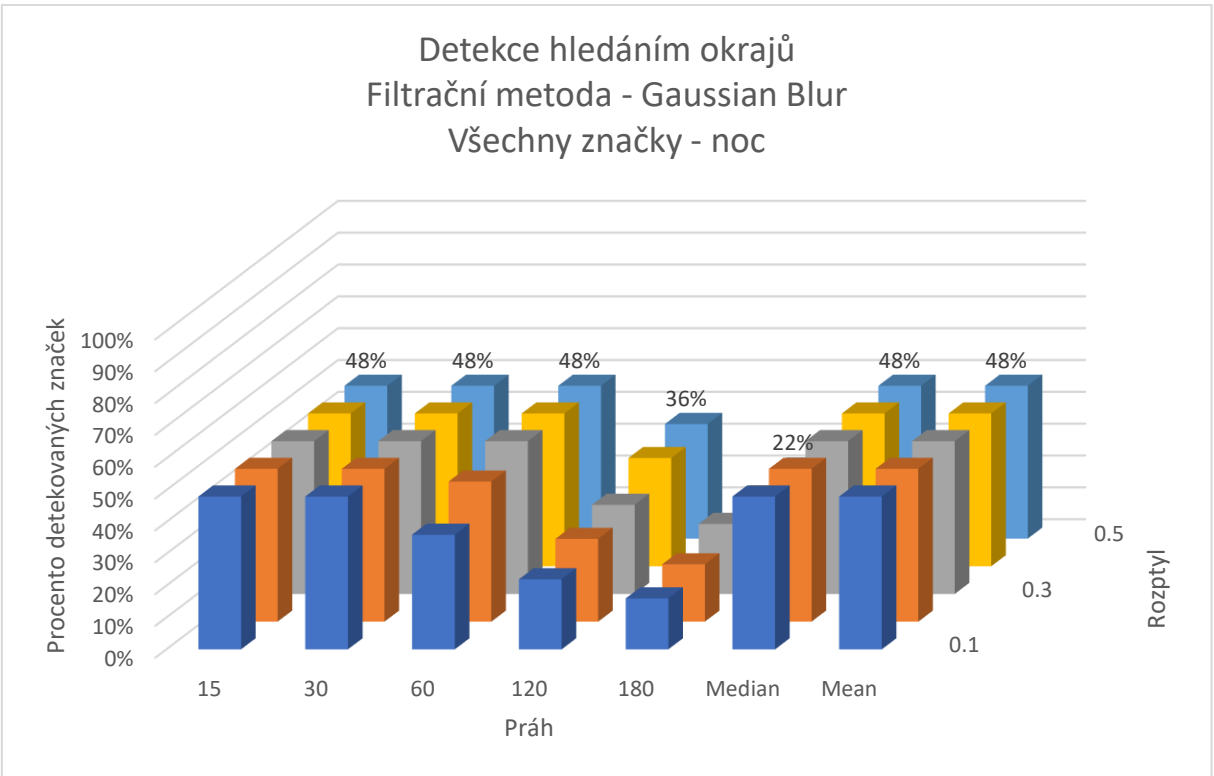
Obr. 7.11 Průběh detekcí hledáním okrajů pro slunečno pomocí bilaterálního filtru



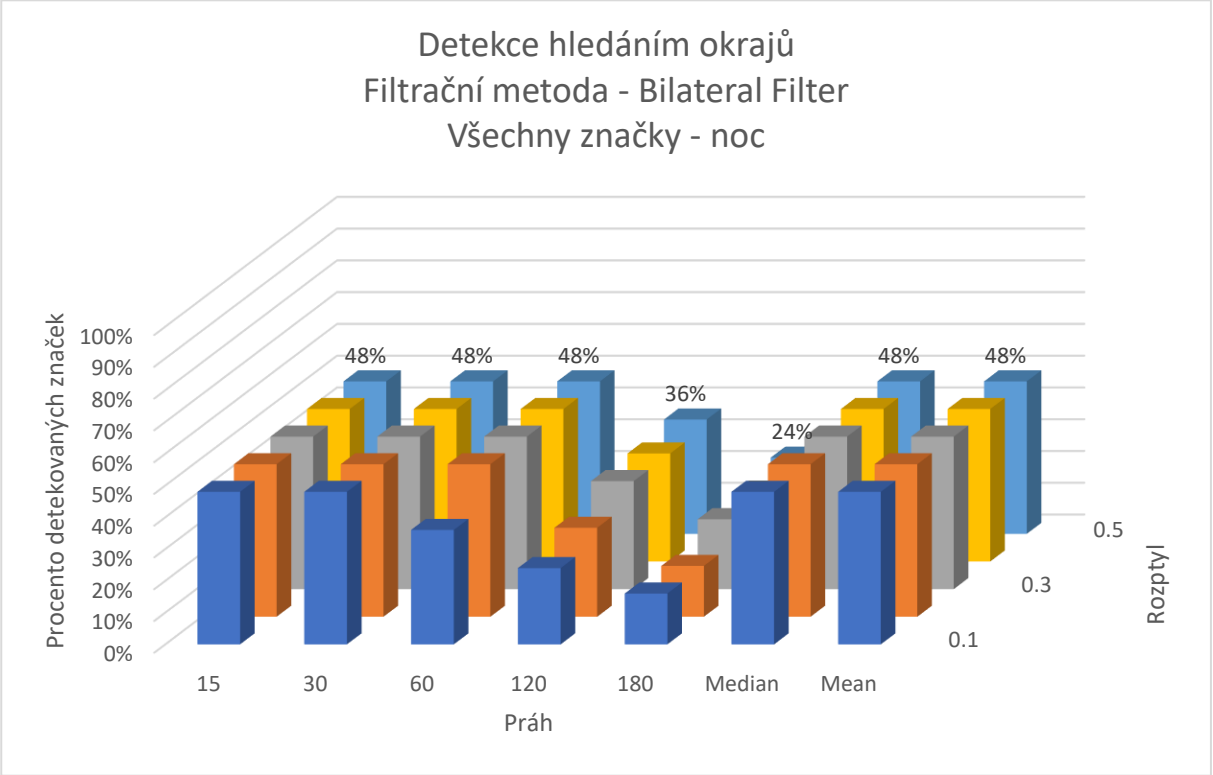
Obr. 7.12 Průběh detekcí hledáním okrajů pro oblačno pomocí Gaussova filtru



Obr. 7.13 Průběh detekcí hledáním okrajů pro oblačno pomocí bilaterálního filtru



Obr. 7.14 Průběh detekcí hledáním okrajů pro noc pomocí Gaussova filtru



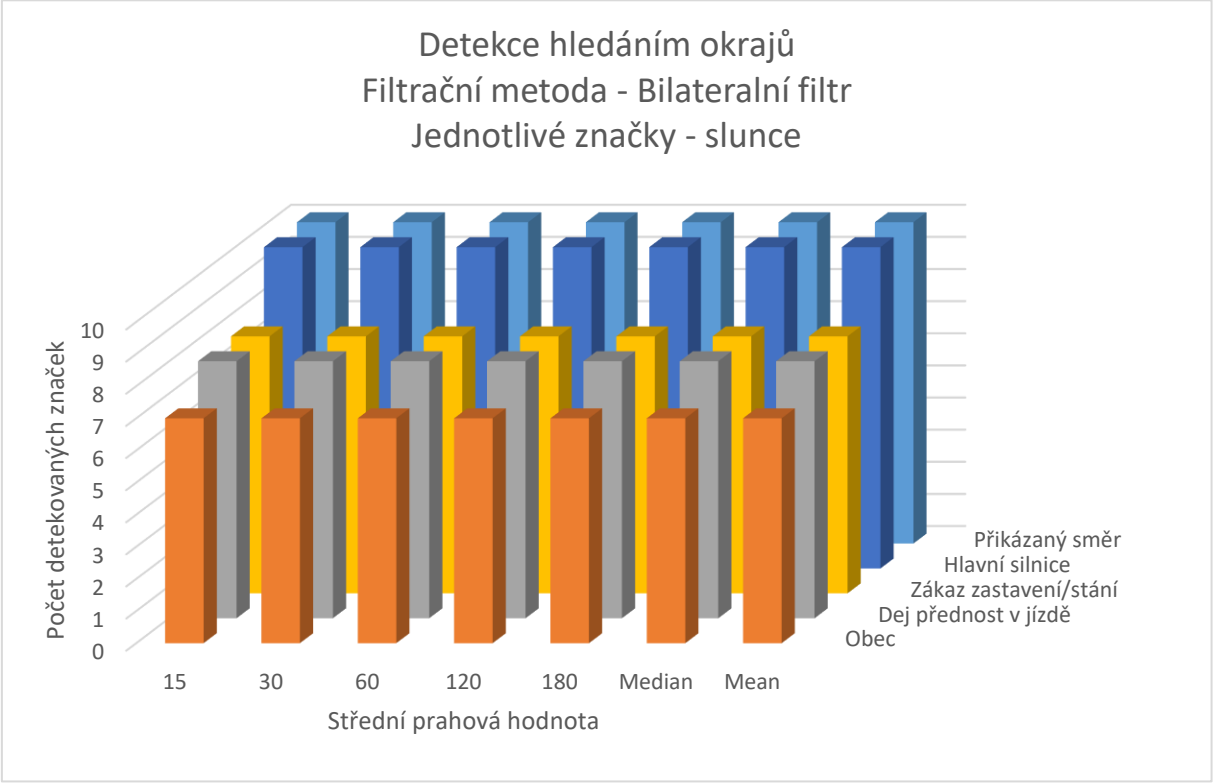
Obr. 7.15 Průběh detekcí hledáním okrajů pro noc pomocí bilaterálního filtru

Z těchto dílčích výsledků vyplývá, že jejich průběhy detekcí se příliš neliší od těch celkových. Podle denní situace se mění pouze množství detekovaných značek. Bilaterální filtr pak opět vykazoval lepší výsledky mimo oblačného počasí.

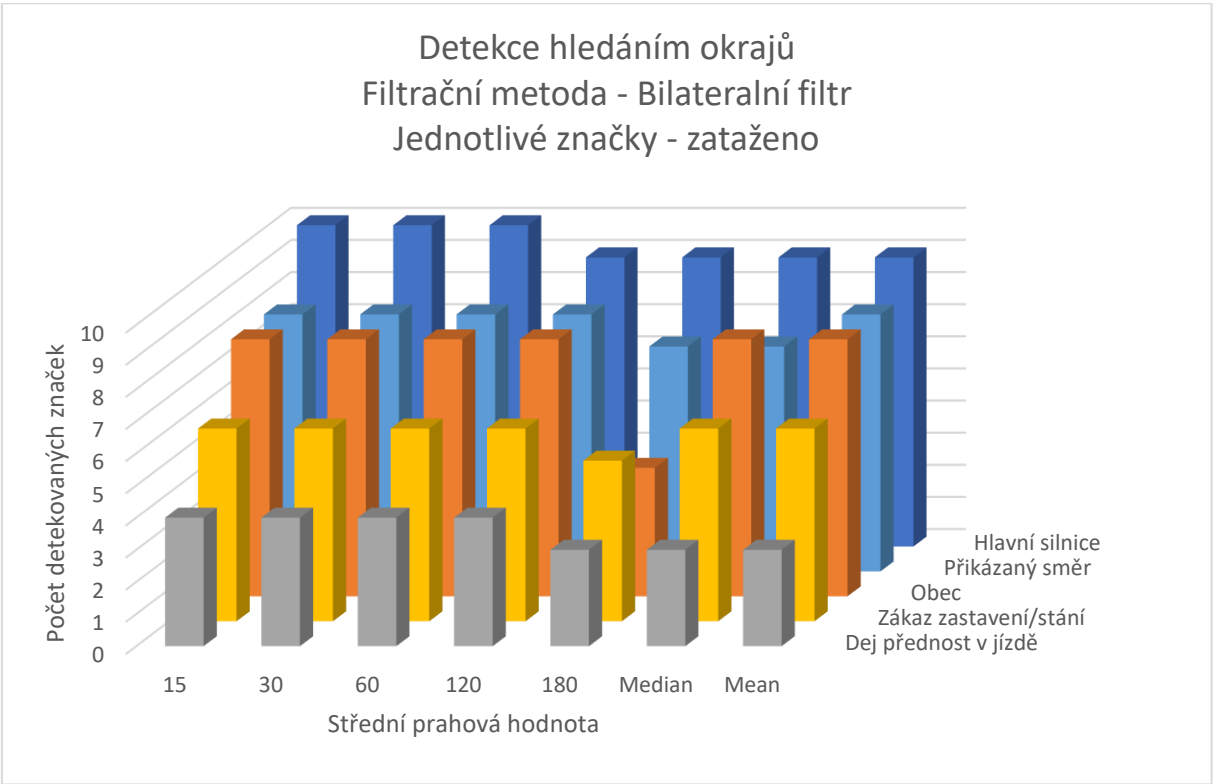
Detekce ve slunečném počasí ukázala, že vůbec nezáleží na jejím nastavení a množství detekovaných značek je vždy stejné. Mění se pouze množství falešně pozitivních detekcí. Celková schopnost detekce je zde ale až 86 %. Pro „zataženo a noc" zůstává průběh množství detekovaných značek stejný opět pouze do střední prahové hodnoty „60" a poté postupně klesá. U zataženého počasí je schopnost detekce přibližně stejná jako celková, a to 72 %. Pro noc už pak rapidně klesá až na 48 %.

Obecně lze říct, že tam, kde u středních prahových hodnot množství detekovaných značek klesá, je tento pokles kompenzován velikostí rozptylu, který se zvyšující se hodnotou opět zvyšuje toto množství. Na úkor toho ale opět narůstá počet falešně pozitivních detekcí.

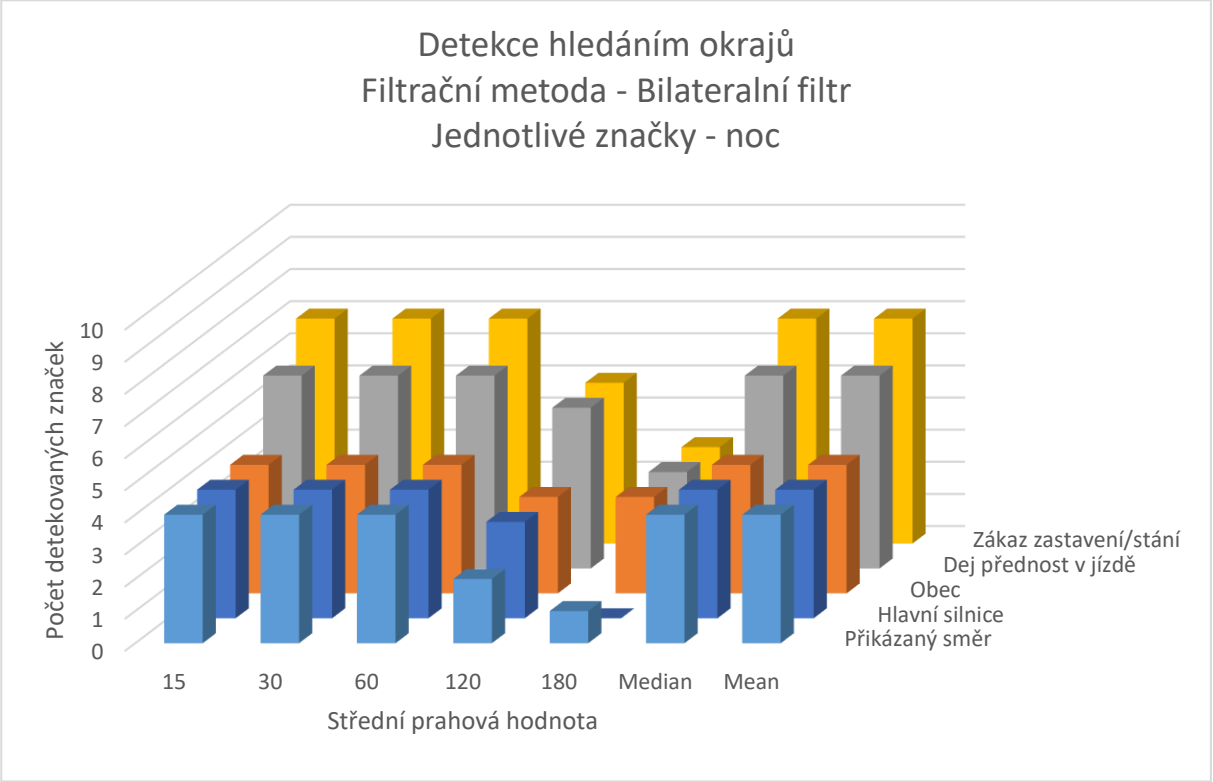
V následujících grafech jsou ještě průběhy detekcí jednotlivých dopravních značek pro porovnání, jak se mění průběh detekce každé značky zvlášť (Obr. 7.16–7.18). Pro vizualizaci se vykreslily výsledky všech značek do jednoho grafu s využitím pouze bilaterálního filtru. Proto se zvolila jen jedna hodnota rozptylu pro každou z nich, a to 0,3. Výsledky pro všechny značky zvlášť a jejich kombinace nastavení jsou v příloze.



Obr. 7.16 Průběh detekcí hledáním okrajů pro jednotlivé značky – Slunce



Obr. 7.17 Průběh detekcí hledáním okrajů pro jednotlivé značky – Zataženo



Obr. 7.18 Průběh detekcí hledáním okrajů pro jednotlivé značky – Noc

Z grafů vyplývá, že množství detekovaných značek je různé pro každý typ značky. Slunečné počasí opět vykazuje celkově největší spolehlivost detekce, kde se navíc u dvou značek dosáhlo 100% úspěšnosti. Nicméně nelze jednoznačně říct, která dopravní značka se nejlépe detekuje, protože se podle denní situace mění různě i množství detekcí pro každou z nich. Lze ale opět predikovat, že pro každou dopravní značku je celkový průběh detekcí v závislosti na nastavení stejný.

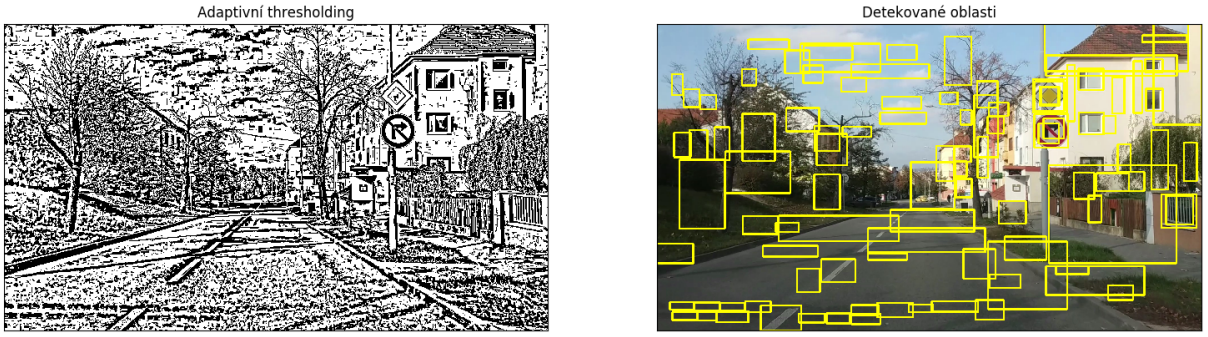
Na základě tohoto shrnutí lze říct, že pro obecné použití mohou být střední prahové hodnoty nastaveny automaticky jako medián i jako průměr pixelů, nebo jako konstanta do maximální hodnoty „60“. V případě menší schopnosti detekce ji lze kompenzovat větší velikostí rozptylu. Je zde ovšem nutné brát v úvahu to, že se pak zvyšuje i počet falešně pozitivních detekcí, jež je nutné následně nějakým způsobem eliminovat. Co se týče volby filtrační metody, je lepší volit bilaterální filtr, protože ve většině případů se při jeho použití dosahovalo největšího množství detekcí. Na druhou stranu má ale delší výpočetní čas, a tak v případě nutnosti snížení tohoto času lze volit i Gaussův filtr.

Celková schopnost detekce je pak v průměru 70 %. Při dílčím použití této metody na jednotlivé denní situace se pak tato schopnost mění. Pro slunečné počasí jí lze dosáhnout až 86 %, pro noc je to ovšem naopak, a to jen 48 %. V případě pro každou značku zvlášť je množství detekovaných značek různé, ale jejich průběhy jsou pořád stejné, a tak lze tuto metodu použít univerzálně, aniž by se muselo měnit nastavení.

7.2 Detekce prahováním obrazu

K detekci pomocí prahování lze použít, tzv. „thresholding“. V knihovně OpenCV má funkce tvar „cv.threshold()“. Pracuje se s černobílým obrázkem, který se upravuje podle zadané prahové hodnoty. V jednoduchém případě se převede na binární obrázek pomocí „binarizace (Python: cv.THRESH_BINARY)“. Pixely nad touto prahovou hodnotou se přepíšou na 1, pod ní zase na 0. Tohle je pro časovou náročnost dobré řešení, protože se pak dále pracuje pouze se dvěma hodnotami. Další možností může být „Threshold to zero (Python: cv.THRESH_TOZERO)“, tedy prahování k nule. Tím se zachová černobílý obrázek a pouze pixely pod prahovou hodnotou se přepíší na 0. Pro oba případy existují i jejich inverzní podoby, které přepisují hodnoty obráceně. Podobně jako prahování k nule funguje i poslední metoda „Truncate (Python: cv.THRESH_TRUNC)“. Ta také zachovává obrázek, ale pixely nad prahovou hodnotou přepisuje jen na její hodnotu. V OpenCV pro tuto metodu ale už inverzní funkce neexistuje [20].

V druhém případě lze použít rozšířenou metodu, a to adaptivní thresholding. Jak již bylo zmíněno, metoda je schopná se přizpůsobit dané denní situaci. To by mělo eliminovat rozdíly při detekci ve dne a v noci. To má ovšem za následek příliš dlouhý výpočetní čas. Protože je nastavení této metody prakticky stejné jako u standardní metody prahování, není nutné ji zahrnovat do experimentálního testování, kde se řeší především spolehlivost při různých nastaveních. Je však vhodné ji ověřit při finálních testech, zda bude schopná lepší detekce než její standardní podoba. Její jediný rozdíl v nastavení, který se netestuje na standardní metodě, je použitý typ. Ten může být s použitím buď průměrné hodnoty, nebo pomocí Gaussova rozložení. Pro porovnání a vizualizaci, jak taková metoda pracuje, jsou následující obrázky (Obr. 7.19–7.20).



Obr. 7.19 Adaptivní thresholding, použití Gaussova rozložení

Obr. 7.20 Adaptivní thresholding, použití průměrné hodnoty

Jak je na obrázcích vidět, metoda dokáže vytvořit obrysy jednotlivých kontur, ovšem to má za následek i vytvoření mnoha falešně pozitivních detekcí. Mimo časově výpočetní náročnost je tohle druhým důvodem, proč se metodou v experimentu prozatím nezabývat, ale soustředit se především na standartní podobu této metody. Výhodou pak je, že výsledky pro jednotlivá nastavení jsou následně aplikovatelná i na toto rozšíření.

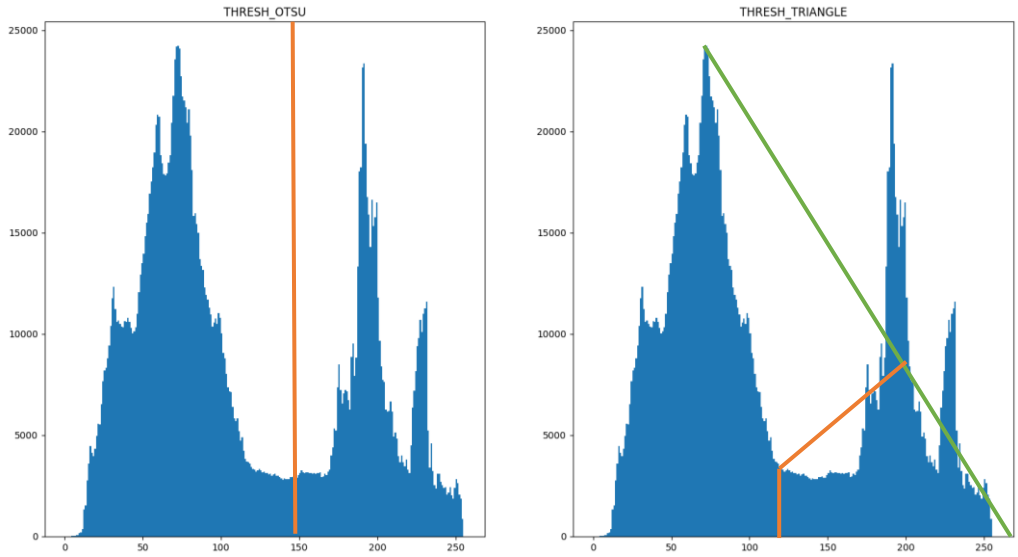
7.2.1 Test metody

V tomto případě není nutné předem upravovat obraz, ale pouze se převede na černobílý, se kterým se dále pracuje. Knihovna OpenCV k tomuto účelu obsahuje jednoduchou funkci „cv.cvtcolor()“, kde je prvním vstupním parametrem obrázek a následně požadavek na převod barev. Je ovšem nutné znát barevný formát obrázku. V tomto případě to je „BGR (Blue, Green, Red)“, a tak výsledný požadavek na převod má tvar „BGR2GRAY“, tedy BGR do Gray (šedá) (Obr. 7.21).

Obr. 7.21 Originální obrázek (vlevo), černobílá verze (vpravo)

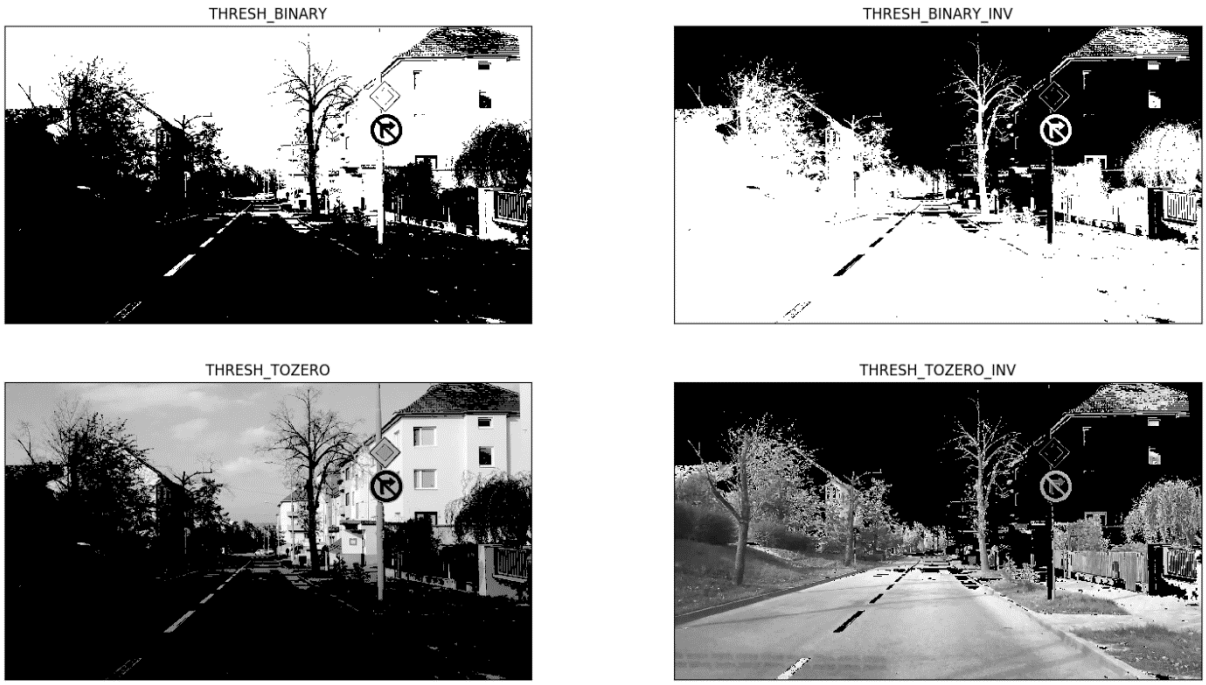
Po této jednoduché úpravě se může přejít k samotné metodě prahování. Testovaly se opět dva parametry pro ověření funkčnosti.

Nejdříve se měnila prahová hodnota, která se postupně zvyšovala od 15 až do 120, tedy přibližně do poloviny maximální hodnoty pixelu (255). Tu je nutné do funkce také zadat, aby věděla, s jakou velikostí pixelu pracuje. Velikost kroku při navyšování byla nastavena pouze na 15, aby byly lépe pozorovatelné změny v množství detekovaných značek pro následné přesnější určení správné prahové hodnoty. Pro automatické určení této hodnoty byly navíc testovány metody „cv.THRESH_OTSU“ a „cv.THRESH_TRIANGLE“, které na základě histagramu daného obrázku určí prahovou hodnotu. Metoda „OTSU“ ji jednoduše určí jako střed mezi dvěma nejvyššími body. Metoda „TRIANGLE“ bere pouze jeden nejvyšší bod, který spojí čarou k nejvzdálenějšímu konci. Maximální kolmá vzdálenost mezi touto čarou a hodnotou histagramu určuje prahovou hodnotu (Obr. 7.22) [21].



Obr. 7.22 Metoda cv.THRESH_OTSU (vlevo) a metoda cv.THRESH_TRIANGLE (vpravo)

Druhým parametrem testování byly výše zmíněné metody prahování, přesněji tedy metody binarizace a prahování k nule, a to jak v jejich standardní, tak i inverzní podobě. Poslední metoda „Truncate" byla odzkoušena, ale protože při prvních testech příliš nefungovala, byla hned z počátku vyřazena z testování. Po této úpravě pak vypadají výsledné obrazy následně (Obr. 7.23).



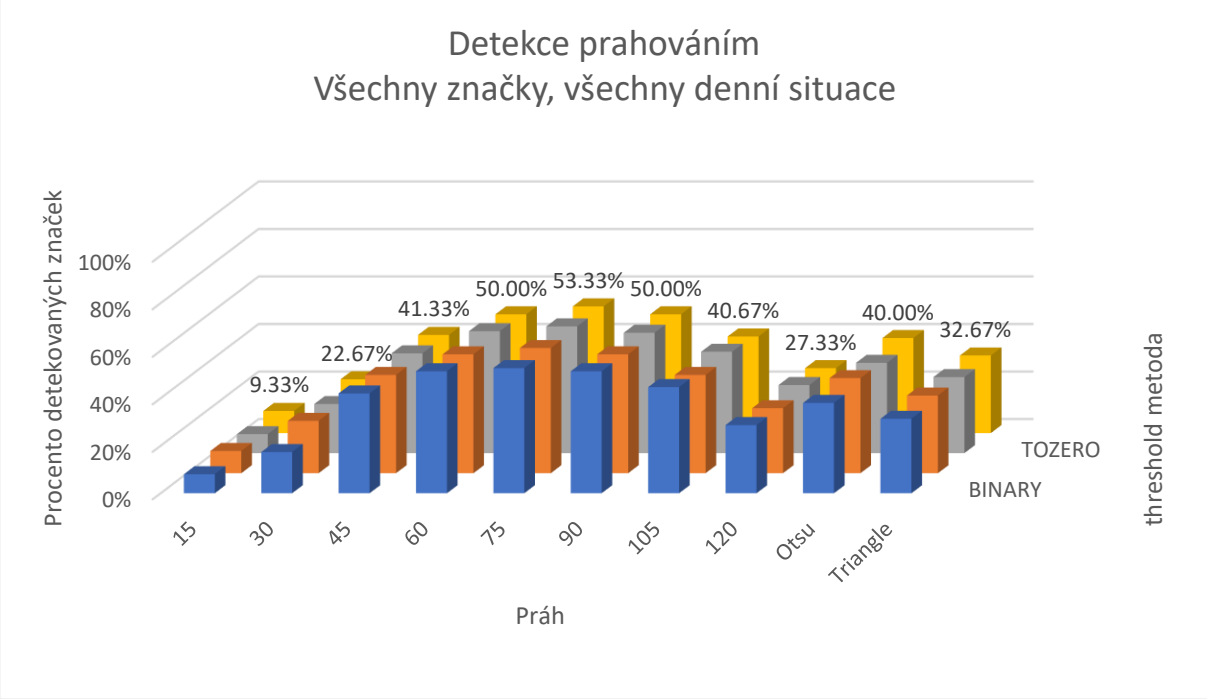
Obr. 7.23 Metody základní (vlevo), metody inverzní (vpravo)

Protože ve výsledném obrazu vzniknou kontury, lze dále postupovat stejným způsobem, jako to je popsané v předešlé metodě hledání okrajů, což znamená provést detekci těchto kontur, odebrání nepoužitelných a ze zbylých pak následně určit oblasti detekce (Obr. 7.24).

Obr. 7.24 Detekované kontury (vlevo) a oblasti detekce (vpravo)

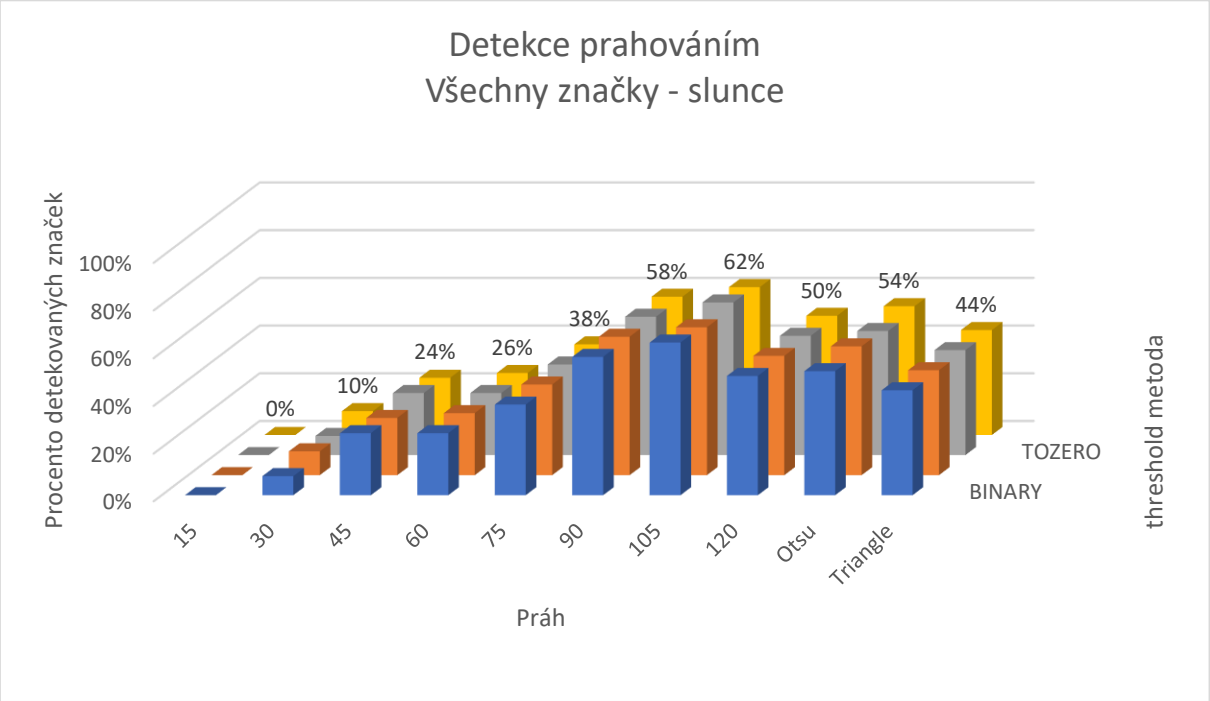
7.2.2 Výsledky

V následujícím grafu jsou shrnuty průběhy detekcí všech značek pro všechna nastavení a všechny denní situace (Obr. 7.25).

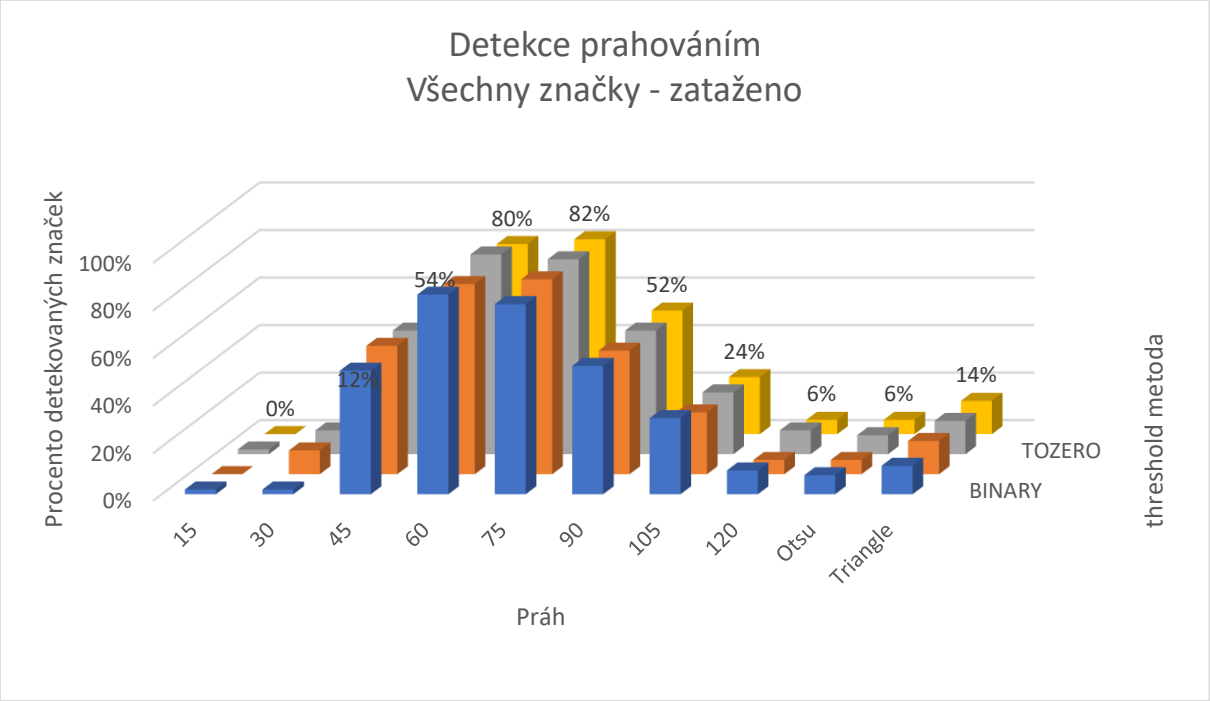


Obr. 7.25 Průběhy detekcí všech značek ve všech denních situacích

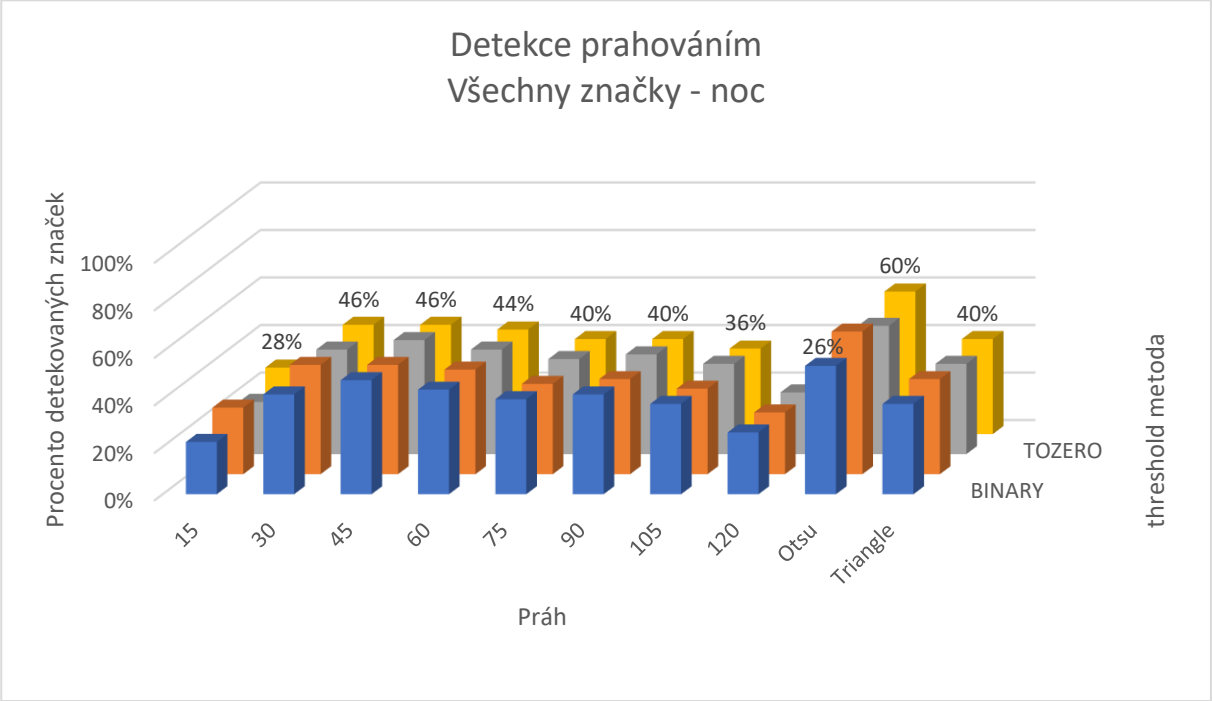
Z grafu lze vidět, že celkové detekce lze dosáhnout pouze s 53% úspěšností. Dále pak, že nezáleží na dané metodě prahování a průběhy zůstávají stále stejné. Celkový čas detekce se pohyboval přibližně do 40 ms. Další výsledky se ale jen těžko vyvozují, a proto se tyto průběhy rozvedly na jednotlivé denní situace (Obr. 7.26–7.28).



Obr. 7.26 Průběhy detekcí všech značek – Slunce



Obr. 7.27 Průběhy detekcí všech značek – Zataženo



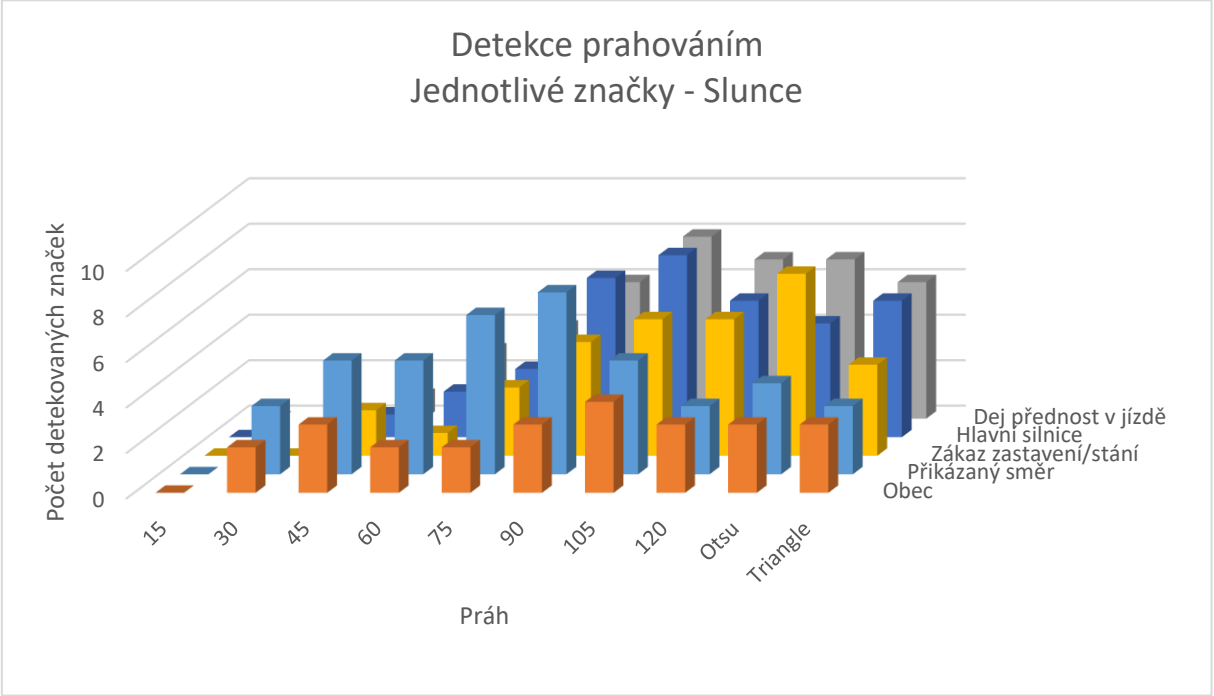
Obr. 7.28 Průběhy detekcí všech značek – Noc

U těchto výsledků lze už vidět velké rozdíly v průběhu pro jednotlivé denní situace. Výsledná detekce se tak pro každou z nich zvýšila. Na to má ovšem ale vliv to, že každá denní situace vyžaduje různé nastavení, jak je vidět na jednotlivých grafech. Metoda prahování opět příliš nemění celkový průběh, a tak na ni nezáleží. Z hlediska časové a pamětové náročnosti je ale lepší volit metodu binarizace kvůli tomu, že se pak pracuje pouze se dvěma hodnotami. Vzhledem k dílčím výsledkům, především v noci, se pak jeví její inverzní podoba jako ta spolehlivější.

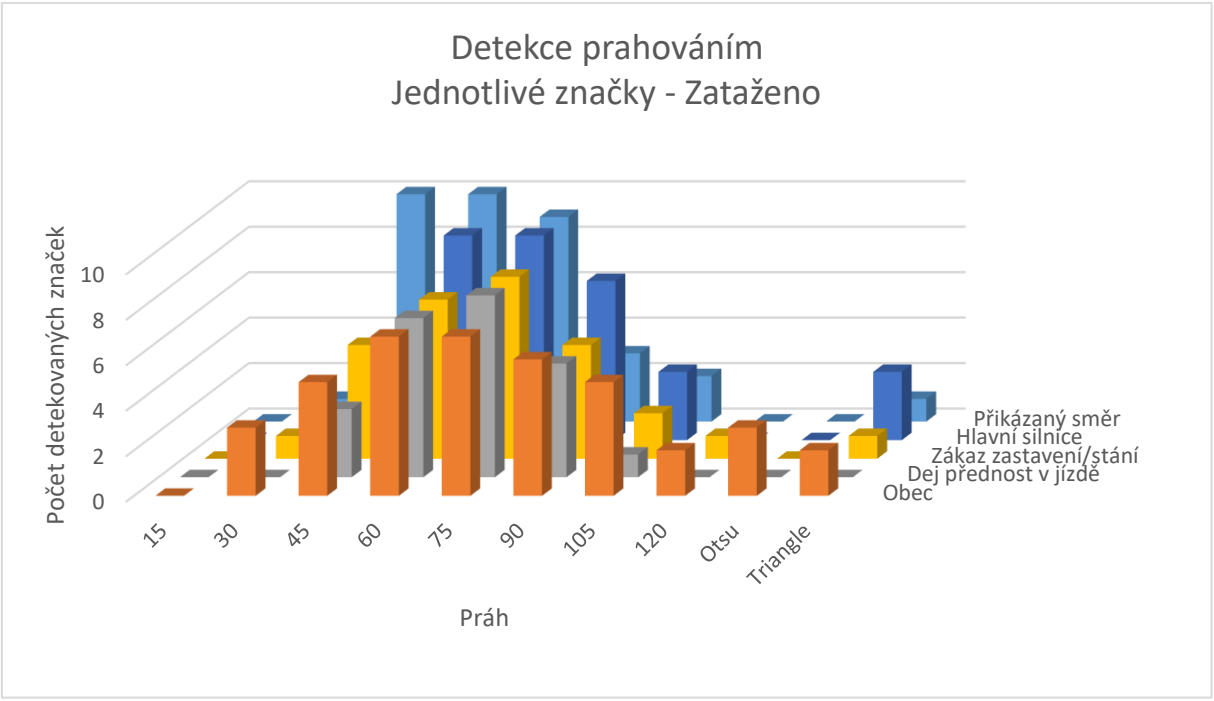
Obecně lze shrnout, že pro slunečné počasí dosahuje maximální úspěšnosti prahová hodnota „105" s výsledkem 62 %. Může se brát v úvahu i metoda automatického rozpoznávání prahové hodnoty „cv.THRESH_OTSU", která sice dosahuje už o něco menší úspěšnosti, ale pořád je jedna z nejvyšších, a to 54 %. Pro situaci „zataženo" se pak maximální procento úspěšnosti posouvá k prahové hodnotě „75", kde dosahuje až 82 %. Na druhou stranu zde ale přestává téměř úplně fungovat automatické rozpoznávání prahové hodnoty. Testování značek pro „noc" ukázalo, že touto metodou nelze detekovat ani polovinu značek při pevném nastavení prahové hodnoty „30–45", kde je maximální úspěšnost jen 46 %. Nejlepších výsledků dosahuje ale automatické rozpoznávání pomocí „OTSU" metody, kde lze dosáhnout úspěšnosti až 60 %.

Lze tedy říct, že metoda může dosáhnout celkem spolehlivých výsledků, ovšem je nutné nejdříve zajistit různá nastavení prahových hodnot. Lze ale částečně predikovat, že se tyto hodnoty snižují se snižováním světelné intenzity, jak je vidět na grafech.

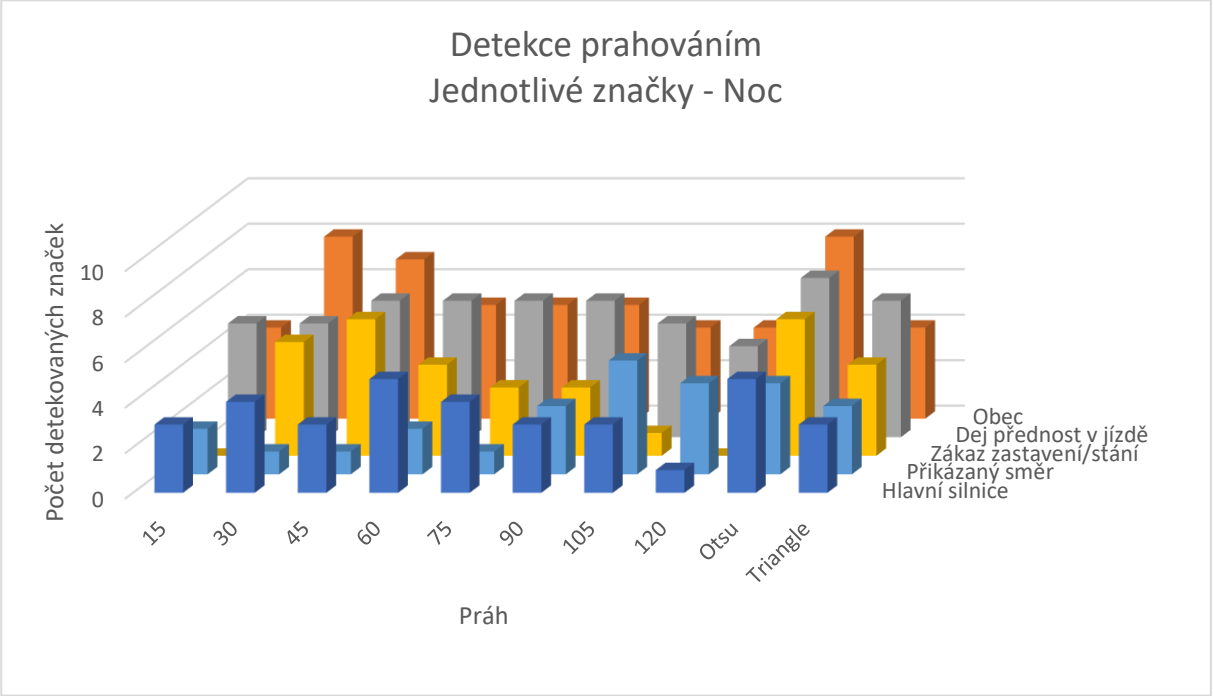
V následujících grafech jsou ještě znázorněny průběhy detekcí pro každou značku zvlášť (Obr. 7.29–7.31). Pro vizualizaci se znázornil průběh detekce pouze s použitím binární inverzní metody. Všechny průběhy jsou pak znázorněny v příloze.



Obr. 7.29 Graf průběhů detekcí jednotlivých značek – Slunce



Obr. 7.30 Graf průběhů detekcí jednotlivých značek – Zataženo



Obr. 7.31 Graf průběhů detekcí jednotlivých značek – Noc

Z těchto grafů lze vyčíst, že ačkoliv má každá značka prahovou hodnotu maximální detekce vždy jinde, není to příliš velký rozdíl a jejich průběhy odpovídají přibližně celkovým průběhům jednotlivých denních situací. Z toho lze usoudit, že není potřeba pro každou značku speciální nastavení a metodu lze použít univerzálně alespoň pro jednu vybranou denní situaci. Výjimkou je ovšem noc, kdy to nelze jednoznačně určit, a lze tedy jen spoléhat na automatické rozpoznávání prahové hodnoty.

Ze všech těchto poznatků vyplývá, že detekce pomocí prahování obrázku je celkem spolehlivá a lze s její pomocí dosáhnout určitého vysokého stupně detekce, ale pouze za předpokladu, že se zajistí rozdílná nastavení pro každou značku, nebo alespoň denní situaci. To by vedlo k nastavování prahových hodnot například podle intenzity osvětlení, kdy se snižující se touto intenzitou se snižuje i požadovaná prahová hodnota. K tomu by ovšem bylo nutné vymyslet, jak tuto intenzitu měřit.

Obecně tedy dosahuje metoda celkové detekce pouze 53 %. Při rozdělení na jednotlivé denní situace lze dosáhnout o něco vyšších detekcí, ovšem při zajištění správného nastavení pro danou situaci. Při slunečném počasí a v noci je schopnost detekce přibližně 60 %. V oblačném počasí až 82 %.

7.3 Detekce segmentací barev

Každý pixel s sebou nese hodnotu o jeho barvě. Ta závisí na formátu, ve kterém je obrázek vygenerován. Mezi základní patří například formát RGB. Ten obsahuje hodnoty červené, zelené a modré barvy. Jejich různou kombinací pak vzniká barva nová. Nulové hodnoty vytváří černou, naopak maximální hodnoty zase bílou barvu. Dalším formátem může být například HSV, který odpovídá lidskému vnímání barev. Jeho složky se dělí na

odstín barvy, sytost a jas. Každá tato složka má své vlastní rozmezí, ve kterém se může pohybovat. Nesmí však docházet k nesmyslným kombinacím.

Detekce se pak provádí segmentací jednotlivých barev, typických pro dopravní značky. Jsou to především ty, které je ohraničují, aby pak oblast detekce zahrnovala celou značku. Snímek je tak převeden na binární obraz, že se hodnoty hledané barvy v jejich určitém rozmezí převedou na 1, zbytek pak na 0.

7.3.1 Test metody

Velkou nevýhodou této metody je, že pro každou denní situaci se mění osvětlení a každá značka je tak vždy jinak nasvícená. Tím se mění odstín její barvy. Proto byl pro testování zvolen formát barvy HSV, kde jsou tyto složky snadněji nastavitelné.

Nejdříve bylo nutné určit hodnoty jednotlivých barev, s jejichž pomocí se budou detekovat značky. K tomu se využil algoritmus, kde se klikáním levého tlačítka myši na obrázku získají hodnoty barev v dané pozici. Zprůměrováním těchto hodnot se určí přibližná hodnota barvy. K ní se pak přidá rozptyl, který ve výsledku určí prahové hodnoty dané barvy. S jejich pomocí se vytvoří maska, tedy binární obraz po segmentaci (Obr. 40–42). Protože se v různých pozicích měnily jednotlivé hodnoty barevného formátu a tím i vytvořená maska, sledováním změn lze pak odhadnout mimo hodnoty dané barvy také velikost rozptylu. Čím větší byly rozdíly v jednotlivých složkách pixelů, tím větší se zde volil rozptyl. Doladění těchto hodnot probíhalo již zcela intuitivně podle průběžných celkových výsledků. Výsledné hodnoty pro jednotlivé denní situace i obecná hodnota pro všechny jsou vypsány v následující tabulce (Tabulka 7.1).

Barva	Denní doba	H	S	V
Červená	Den	175±10	210±70	125±80
	Zataženo	177±10	137±30	84±80
	Noc	5±10	95±30	70±80
	Obecné	175±20	158±60	118±120
Modrá	Den	109±15	205±30	71±100
	Zataženo	109±15	188±30	69±100
	Noc	112±15	154±30	48±100
	Obecné	110±15	182±60	63±120
Bílá	Den	63±40	21±55	185±15
	Zataženo	62±40	11±55	117±20
	Noc	30±40	5±55	98±20
	Obecné	40±65	14±60	143±30

Tabulka 7.1 Hodnoty barevného formátu HSV a jejich zvolený rozptyl

K odfiltrování barev na binární obraz slouží v OpenCV funkce „cv.inRange()“, do které se zadává požadovaný rozptyl hodnot barev. Jako ty nejběžnější se prozatím detekovaly červená, modrá a bílá barva (Obr. 7.32–7.34).

Obr. 7.32 Separování červené barvy

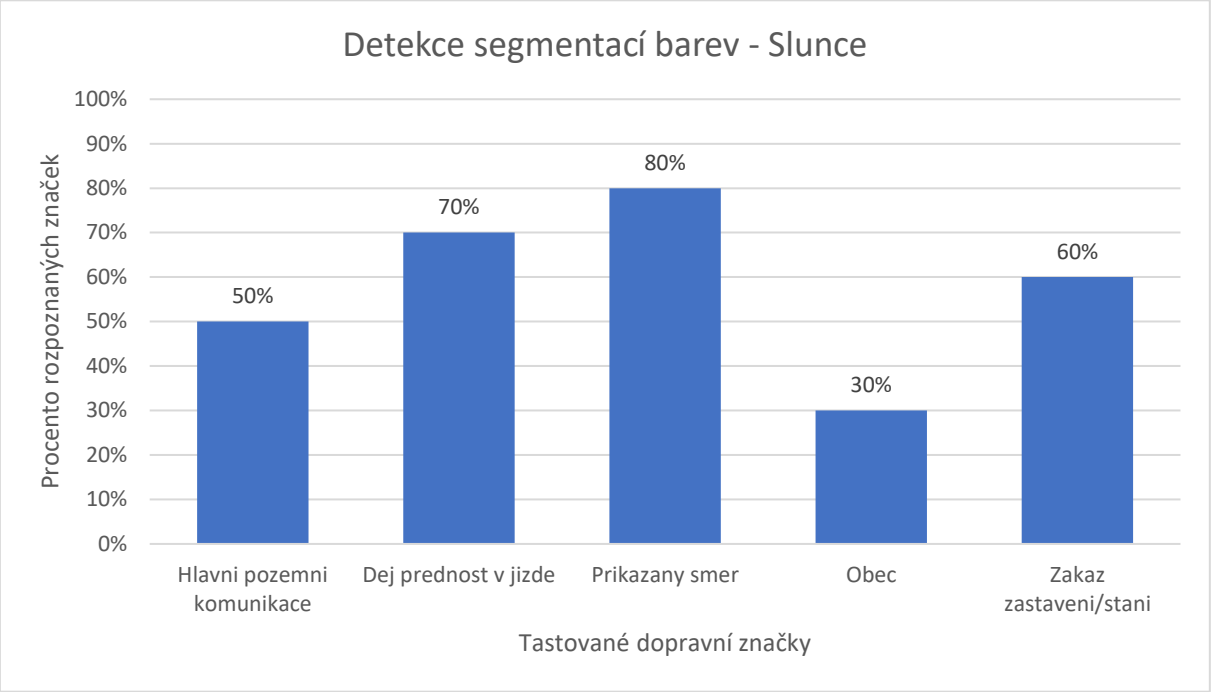
Obr. 7.33 Separování modré barvy

Obr. 7.34 Separování bílé barvy

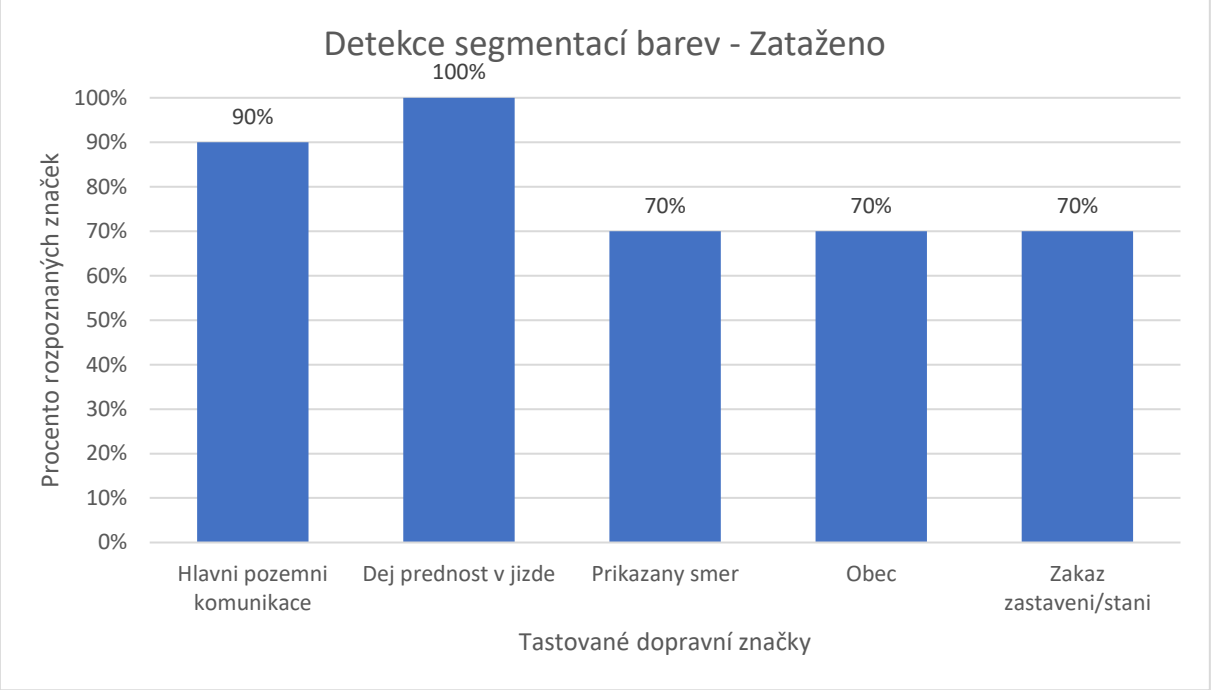
Protože jsou opět z binárního obrazu detekovatelné kontury, další postup je úplně stejný, tedy detekce těchto kontur, odebrání nepoužitelných a nakonec ze zbylých určit oblasti detekce.

7.3.2 Výsledky

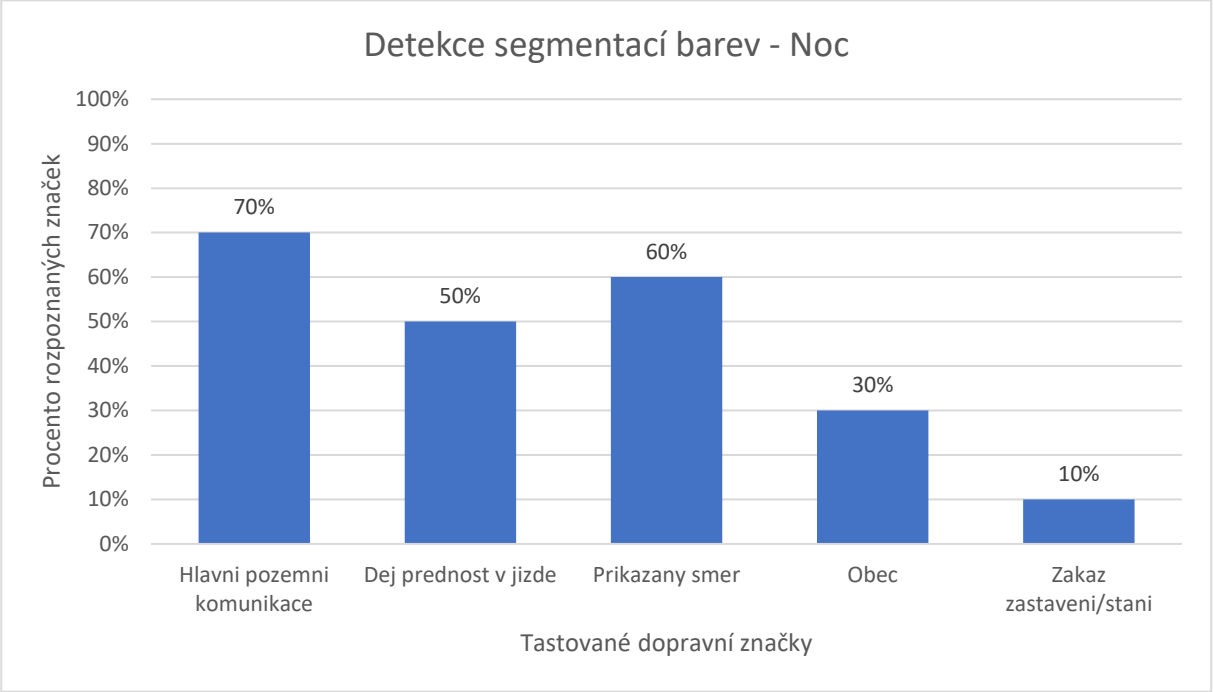
Výsledky detekce pro jednotlivé značky s předem nastavenými hodnotami jednotlivých barev a jejich rozptylu pro jednotlivé denní situace, jsou v následujících grafech (Obr. 7.35–7.37).



Obr. 7.35 Metoda detekce separováním barev – Slunce

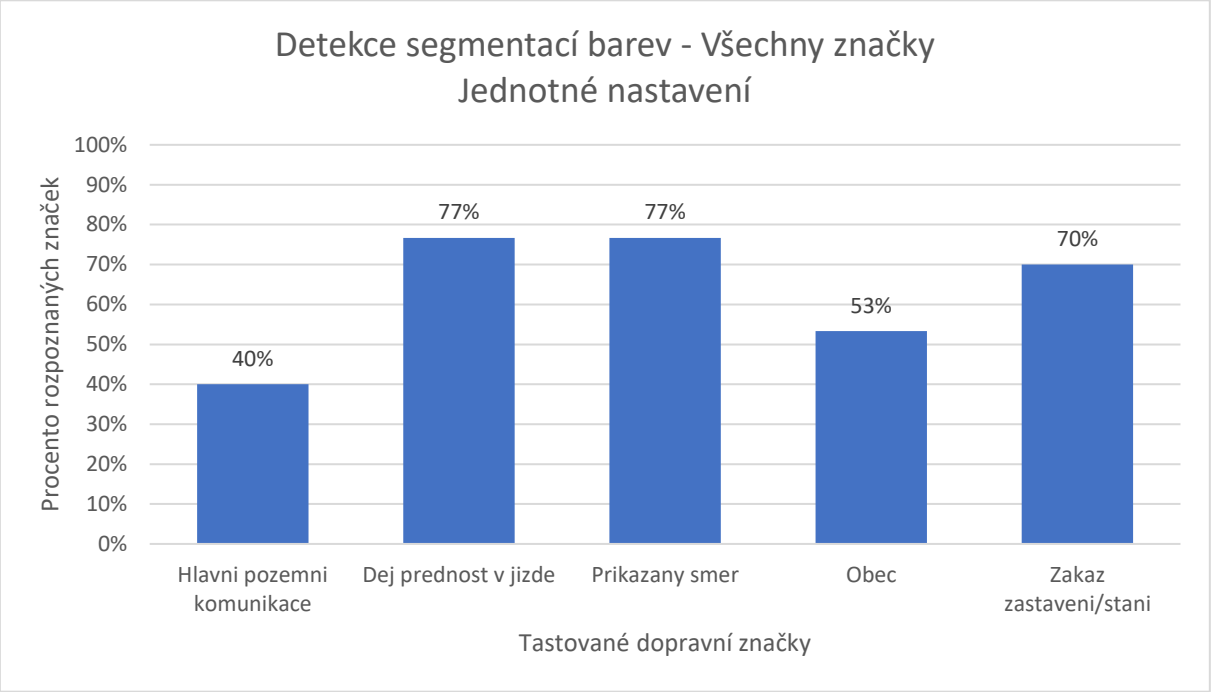


Obr. 7.36 Metoda detekce separováním barev – Zataženo



Obr. 7.37 Metoda detekce separováním barev – Noc

Z grafů je vidět, že pro jednotlivé značky je schopnost detekce různá v každé denní situaci, a to i navzdory, že jsou vždy nastaveny přesné hodnoty dané barvy pro konkrétní situaci. Sjednocením do jednoho grafu nelze množství detekovaných značek pouze sečíst a určit tak poměr s celkovým množstvím testovaných značek, jako v předchozích případech. Je to proto, že jsou zde hodnoty jednotlivých barev různě nastavené. Pro určení celkové schopnosti detekce se tedy musely znova provést všechny testy s obecným nastavením těchto hodnot. To se získalo jejich zprůměrováním a následným doladěním podle průběžných výsledků. Množství všech nově detekovaných značek s obecným nastavením barevného formátu pro všechny denní situace jsou v následujícím grafu (Obr. 7.38).



Obr. 7.38 Metoda detekce separováním barev – Všechny denní situace

Při použití jednotného nastavení pro všechny denní situace lze vidět u některých značek nárůst v množství detekovaných značek, ovšem u jiných naopak zase pokles. Obecně se průměrná hodnota všech detekovaných značek pohybuje okolo 63 %.

I když se metoda zpočátku nejevila jako příliš vhodná, testování ukázalo, že v určitých případech je schopná vysoké schopnosti detekce. Je to ovšem dané tím, že barevný formát se u každé značky mění vždy podle její barvy a aktuální denní situace. Při nastavení obecných hodnot již schopnost detekce převážně klesá. Detekci je navíc nutné provádět pro každou barvu zvlášť. Pro jeden snímek se tedy musí provést minimálně tři různé detekce. To vede na časově delší výpočet, který se pohybuje od 50 ms do 100 ms, a samozřejmě opět také k nárůstu falešně pozitivních detekcí.

8. Rozpoznávání tvarů

Protože při samotné detekci dopravních značek vzniklo mnoho falešných detekcí, musí se před použitím rozpoznávacích metod co nejvíce eliminovat. Tato eliminace lze provést samozřejmě i samotným rozpoznáváním značek v detekovaných oblastech, zjištěním malé shody se šablonou. Ovšem, jak vyplývá hned z prvního testování metody „matchTemplate", vedlo by to opět k dlouhému výpočetnímu času a snížení rozpoznávací schopnosti. Proto se musí provést eliminace nejdříve na základě jednodušších metod. To urychlí a zpřesní rozpoznávací metody, protože jim bude předkládán minimální počet detekovaných oblastí, ve kterých by se měly nacházet jen značky.

Falešné detekce lze tedy minimalizovat vytříděním všech detekovaných kontur na jednotlivé tvary. Je to proto, že dopravní značky se oproti okolí vyznačují právě svým tvarem. Velkou výhodou je navíc to, že se detekované kontury roztřídí podle svých tvarů, a tak lze pro následné rozpoznávání použít pouze šablony s tímto tvarem. Tím se zmenší jejich počet pro každou oblast a zkrátí se výpočetní čas.

Z detekovaných kontur lze rozpoznávat tvary například jejich aproximováním, kde při nastavení určité přesnosti z nich vzniknou jednotlivé tvary. Výsledný počet stran pak určuje, o jaký tvar se jedná. Lze tak tedy teoreticky rozpoznávat až čtyři tvary, a to trojúhelník, obdélník, osmiúhelník a kruh. Kruh se pak určuje například jako devět a více stran, což ovšem může zahrnovat mimo to i mnoho jiných tvarů, a nedojde tak k jejich eliminaci. Metoda tak ztrácí význam. Z toho důvodu je lepší přistoupit k jinému způsobu rozpoznávání tvarů, a to k porovnávání kontur se šablonou pomocí korelace. Je to vlastně stejný způsob, který využívá metoda „matchTemplate" k finálnímu rozpoznávání značek, ale v tomto případě lze porovnávat pouze binární obrázky s daleko menším počtem šablon.

8.1 Rozpoznávání pomocí korelace

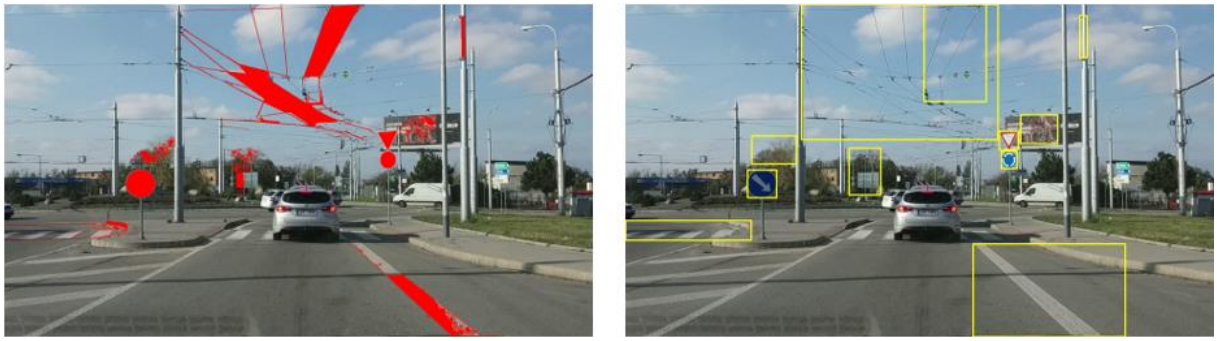
Jak již bylo zmíněno, korelační metoda porovnává detekovaný tvar se šablonou. Proto je potřeba nejdříve vytvořit tyto šablony pro všechny možné tvary. Protože se jedná o korelaci, záleží zde i na natočení daného tvaru, a proto se musí vytvořit šablony pro všechny tyto možnosti. Celkem se tedy jedná o šest různých tvarů, vytvořených do binární podoby o velikosti 50×50 px (Obr. 8.1).

Obr. 8.1 Šablony tvarů dopravních značek pro korelaci

Pro každý tento tvar tedy existuje alespoň jedna značka. Velkou výhodou ale je, že hned pro tři tyto tvary existuje právě jen jedna značka, která je tím unikátní. Díky tomu lze při detekci tohoto tvaru s velkou pravděpodobností říct, že se jedná o danou značku, a následně lze například i snížit nároky na korelační shodu při jejím rozpoznávání, protože

se porovnává pouze s jednou šablonou, a tak jde jen o její ověření. Přesněji se tedy jedná o značky „Dej přednost v jízdě“, „Hlavní pozemní komunikace“ a „Stop“.

Po vytvoření šablon se v dalším kroku musí upravit detekované kontury, tak aby měly stejnou podobu a bylo možné je porovnávat. Použitím některé z metod detekce lze zjistit pozice jednotlivých bodů kontur a velikost detekované oblasti ve které se nachází (Obr. 8.2).



Obr. 8.2 Detekované kontury (vlevo), oblasti detekce (vpravo)

Podle velikosti detekované oblasti se následně vytvoří stejně velké „bílé“ pole (pole s hodnotami pixelu 255). Do něj se pak vykreslí všechny body kontur včetně jejich výplně. Pokud se vykreslí „černě“ (hodnota pixelu 0), vznikne binární pole s detekovaným tvarem. Po jeho úpravě na stejnou velikost šablony, tedy 50×50 px, vypadají detekované tvary kontur následně (Obr. 8.3).

Obr. 8.3 Tvary některých detekovaných kontur

Na takto vzniklé tvary lze již následně aplikovat korelaci pro porovnání se šablonami a určení daného tvaru. Lze k tomu použít předepsanou funkci „matchTemplate“, která používá několik typů korelace, nebo vzhledem k tomu, že se jedná o stejné velikosti, se může použít vlastní korelační metoda podle následujícího vzorce (rovnice 3):

$$r = \sum_{i=0}^{w}\sum_{j=0}^{h} \frac{\left(f(x_i,y_j) - f_{mean}\right) * \left(g(x_i,y_j) - g_{mean}\right)}{\sqrt{\left(f(x_i,y_j) - f_{mean}\right)^2} * \sqrt{\left(g(x_i,y_j) - g_{mean}\right)^2}} * 100 \qquad (3)$$

kde:

f – pole hodnot detekovaných kontur

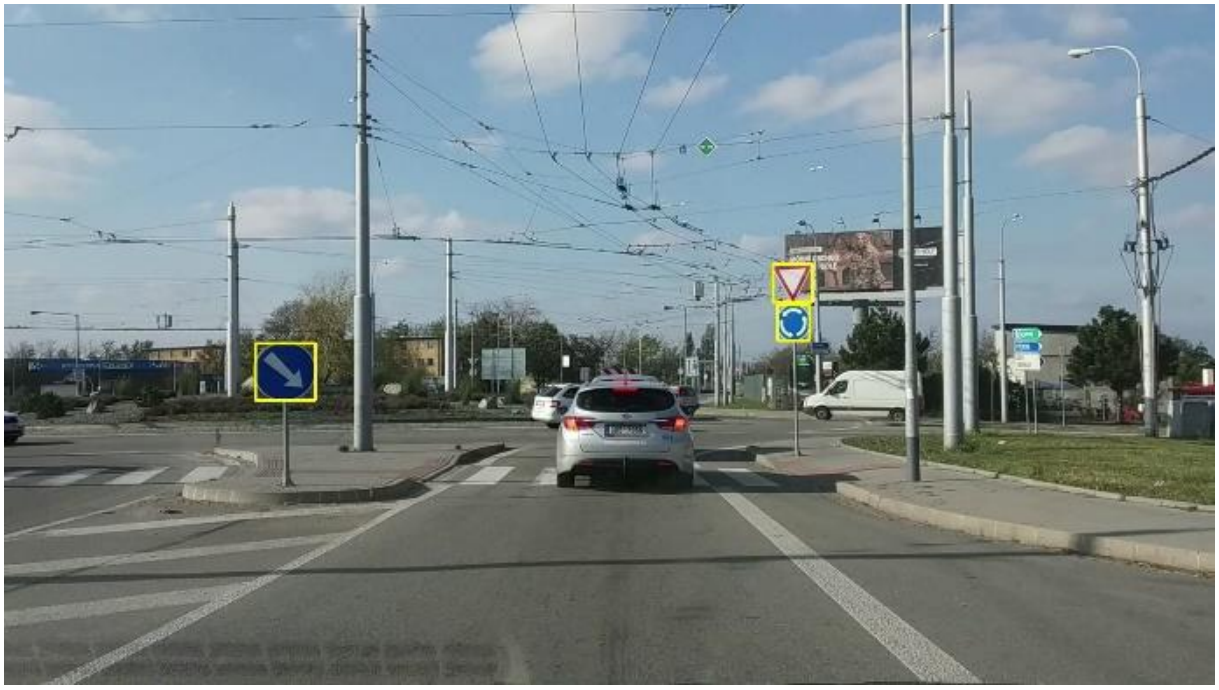
g – pole hodnot šablony

f_{mean} – průměrná hodnota pole detekovaných kontur

g_{mean} – průměrná hodnota pole šablony

r – korelační koeficient

Korelační koeficient zde značí procento shody mezi těmito dvěma obrazy. Při dosažení tohoto koeficientu nad určitou a předem definovanou mez je daný tvar rozpoznán. Pokud tuto mez přesáhne více koeficientů, musí se dále počítat se všemi možnostmi detekovaných tvarů. Toto lze částečně ovlivnit zvýšením meze shody, avšak čím vyšší bude, tím hůře se budou rozpoznávat nepřesné tvary (natočené, useknuté, …). Zbylé tvary kontur jsou pak definovány jako falešně pozitivní detekce a jsou odebrány. Výsledný obraz detekovaných oblastí vypadá následně (Obr. 8.4).



Obr. 8.4 Oblasti detekce po eliminaci falešných detekcí

V tomto případě se tedy povedlo detekovat všechny dopravní značky v obraze a úplně eliminovat falešně pozitivní detekce při nastavení korelačního koeficientu na 70 %. Při experimentování se jevilo nastavení tohoto koeficientu na 70–80 % jako optimální řešení. Finální nastavení bude záviset až na celkovém testování celé aplikace.

9. Rozpoznávání

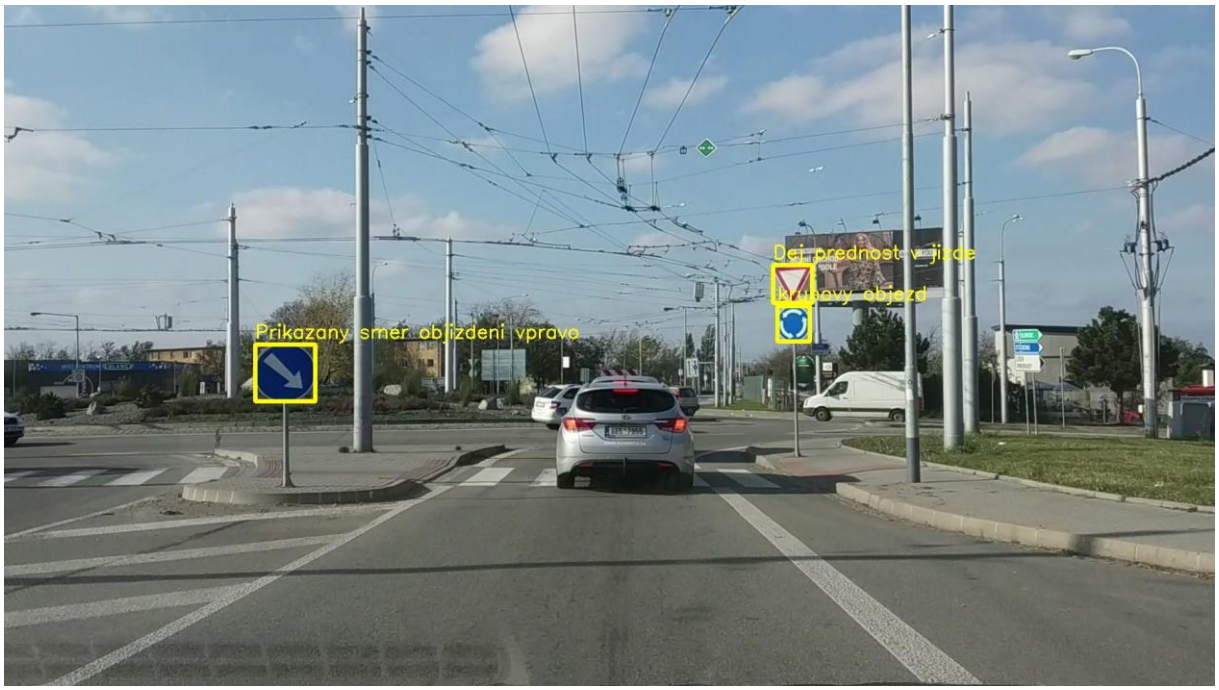
Samotné rozpoznávání vychází tedy z již detekovaných oblastí. Na ty se po předchozích úpravách už může opět aplikovat korelační metoda „matchTemplate“. V případě falešně pozitivních detekcí, které se nepodařilo eliminovat v předešlém kroku rozpoznávání tvarů, se musí tato eliminace provést už jedině v tomto posledním kroku.

9.1 MatchTemplate

Tahle metoda byla popsána již dříve, kdy se používala přímo k detekci s rozpoznáváním. Teď se jedná o její opětovné použití přímo na detekované značky. Výpočet je tím mnohonásobně urychlen. Detekované značky se pak zmenšují, nebo zvětšují na velikost šablony. Ta zůstává konstantní kvůli zachování stejného počtu pixelů a tím i výpočetního času pro všechny rozpoznávací metody.

9.1.1 Test metody

V tomhle případě je metoda rozšířena o více šablon různých typů značek a snaží se tak najít tu správnou hledáním nejlepší korelační shody. Obecně byly vybrány všechny dopravní značky, se kterými se lze běžně setkat na silnici. Jedná se tak o většinu výstražných, příkazových, zákazových dopravních značek a značek upravujících přednost. Naopak se ve většině případů vyloučily všechny informativní dopravní značky, dodatkové tabulky a další. Se zvyšujícím se počtem těchto šablon se ovšem zvyšuje i pravděpodobnost, že může vzniknout vyšší korelační shoda u jiné značky, než je právě detekována. Proto bylo nejdříve nutné vyzkoušet opět všechna nastavení metody, a zjistit tak jejich chování při různých denních situacích. Vzhledem ke stejné velikosti šablony a detekované oblasti se pro porovnání testy rozšířily o vlastní korelační metodu stejnou jako při rozpoznávání tvarů. Výsledný obraz pak vypadá následně (Obr. 9.1).



Obr. 9.1 Detekované a rozpoznané značky v obraze

Jediné, co bylo potřeba upravovat u jednotlivých metod, byla právě mez korelační shody, která udává, zda se jedná o danou značku, či nikoliv, a zároveň také zajišťuje, aby se eliminovaly zbylé falešně pozitivní detekce. Tato mez se nastavovala intuitivně u každé metody od 50 % do 95 %, tak aby pouze eliminovaly falešně pozitivní detekce, a ne hledané dopravní značky. Při detekci některého z unikátních tvarů se pak výrazně snižovala, a to například až k hodnotám 10 %, aby se daná oblast pouze ověřila. Jak již bylo zmíněno, je to z důvodu, že tento tvar s velkou pravděpodobností odpovídá dané značce.

Metoda detekce, která tomu celému předchází, se volila na základě předešlých testů. Hlavním rozhodujícím kritériem byla především celková schopnost detekce. Následně se také z důvodu budoucího možného použití v real-time aplikaci musela uvažovat celková doba jejího výpočtu. Pro porovnání jednotlivých metod mezi sebou je tedy v následující tabulce shrnutí vybraných nejlepších nastavení, jejich spolehlivost a doba výpočtu (Tabulka 9.1).

Metoda detekce	Celková detekce [%]	Slunečno [%]	Zataženo [%]	Noc [%]	Doba výpočtu [ms]
Hledání okrajů – Gausův filtr	68	82	74	48	40–180
Hledání okrajů – bilaterální filtr	68,67	86	72	48	40–180
Prahování obrazu	53,33	62	82	60	40
Separování barev	63,3	58	80	44	50–100

Tabulka 9.1 Porovnání metod detekce

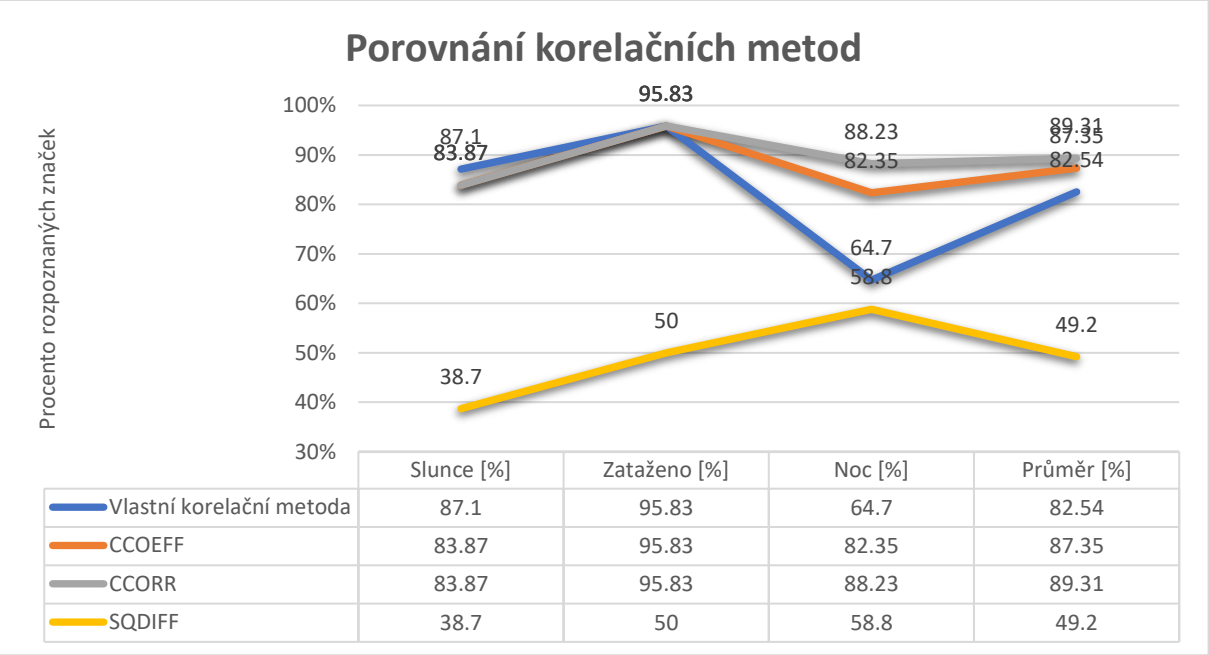
Z tabulky vyplývá, že nejlepší metodou podle celkové schopnosti detekce je metoda hledání okrajů. Ačkoliv v porovnání s ostatními může dosahovat delší výpočetní

doby, a to v krajních případech až 180 ms, celkové množství detekovaných značek je výrazně vyšší. Na druhou stranu je zřejmé, že metoda prahování dosahuje pro jednotlivé denní situace skoro vždy vyšší detekce. Je zde však nutné podotknout, že se v tomto případě jedná vždy o rozdílná nastavení pro každou z nich, která by se musela dodržet. To platí i pro detekci pomocí barev, kde jsou výsledky navíc i o něco horší, co se týká množství detekovaných značek i výpočetního času.

Vzhledem k tomu, že se celkový výpočetní čas u všech metod pohyboval přibližně na stejné hodnotě, a to od 40 ms, má rozhodující vliv pouze celková schopnost detekce. Primárně se tedy pro testování vybrala metoda detekce hledáním okrajů s použitím bilaterálního filtru, který dosahoval o trochu lepších výsledků oproti Gaussovu filtru.

9.1.2 Výsledky

Testy probíhaly teď už pouze v černobílé podobě obrázku, protože doba detekce trvala příliš dlouho. Čas se tak musel částečně kompenzovat, aby se minimalizovala celková doba výpočtu. Výsledky jsou vykreslené v následujícím grafu (Obr. 9.2).



	Slunce [%]	Zataženo [%]	Noc [%]	Průměr [%]
Vlastní korelační metoda	87.1	95.83	64.7	82.54
CCOEFF	83.87	95.83	82.35	87.35
CCORR	83.87	95.83	88.23	89.31
SQDIFF	38.7	50	58.8	49.2

Obr. 9.2 Porovnání korelačních metod pro rozpoznávání

Z výsledného grafu lze určit, že celkový průběh rozpoznávání u všech metod je přibližně stejný, výjimkou je metoda korelace „SQDIFF". Tu není vhodné použít, protože ani jednou nedosáhla lepších výsledků oproti ostatním, ačkoliv její schopnost rozpoznávání postupně stoupá, a to i tam, kde jiné metody vykazují pokles, je zde celkové množství rozpoznaných značek stále malé. Zmíněný pokles u ostatních metod je právě na značkách rozpoznávaných v noci, kde jsou zároveň i největší rozdíly mezi jednotlivými metodami. Problém s nižší spolehlivostí při testování značek v noci je nejen u samotného rozpoznávání, ale i u předešlých metod detekce. Je to následkem absence světla, která znehodnocuje obraz jednotlivých značek. Z toho důvodu je dobré volit vhodnou metodu opět pouze na základě průměrné hodnoty množství rozpoznaných značek.

Nejlepší metodou se podle celkového počtu rozpoznaných značek jeví předepsaná korelační metoda „CCORR“, která dosahuje průměrné schopnosti rozpoznávání téměř 90 %. Je zde však nutné podotknout, že tyto hodnoty množství rozpoznaných značek jsou pouze vzhledem k již detekovaným značkám. Proto je nutné uvažovat k celkovému výpočtu i množství detekovaných značek. To se s použitou metodou hledání okrajů pohybuje přibližně do 69 %. Celkové množství detekovaných a rozpoznaných značek je tak tedy okolo 62 %.

Výsledné množství rozpoznaných značek není sice příliš vysoké, ale tato práce se zabývá především experimentálním porovnáním různých metod zpracování obrazu pro rozpoznávání dopravních značek, čímž se tedy ověřilo, jak dané metody fungují. V tomto okamžiku se navíc jedná pouze o testování různého nastavení na několik testovaných dopravních značek a celkový algoritmus se musí ještě z předešlých poznatků optimalizovat. K tomu se využijí nové testovací vzorky, a to již ve formě videozáznamu, tedy offline testování, kde se budou porovnávat už všechny značky. Optimalizovat by se měla především metoda detekce, která v průměru s necelými 70 % nedosahovala takové úspěšnosti jako metoda rozpoznávání.

Nejdříve se ovšem musejí provést testy s aktuálním nastavením, podle kterých se určí postup optimalizace. Určitým předpokladem je to, že ve videozáznamu je sekvence hned několika snímků za sebou. Tím by mělo být jednodušší alespoň pomocí jednoho z nich určit nebo ověřit danou dopravní značku a zvýšit tak celkovou úspěšnost celého algoritmu.

10. Optimalizace

Na základě poznatků z experimentu byly zvoleny nejvhodnější metody a pro každou z nich se určilo nejoptimálnější nastavení. V tomto případě vyšla nejhůře metoda separováním barev z důvodu obtížnosti nastavení rozdílných prahových hodnot barev při různém nasvícení dopravních značek. Naopak nejvíce vhodnou metodou se pak podle experimentů jevila metoda hledáním okrajů, která dosahovala přibližně stejných průběhů detekce u všech značek bez ohledu na denní situaci. Měnilo se pouze celkové množství detekovaných značek.

Pro nezávislost výsledků se musí použít nové testovací vzorky. Ty byly nasbírány stejným způsobem jako ty předešlé, tedy pořízením videozáznamu jízdy vozidlem pro zachování jejich autentičnosti. Z důvodu snahy o vytvoření real-time systému se metody aplikovaly přímo na vytvořený videozáznam, a ne na vybrané snímky. Tím probíhalo takzvané offline testování. V tomto případě se video pořídilo se standardní vzorkovací frekvencí 30 fps, tedy 30 snímků za sekundu. Pro jejich zpracování je proto nutné, aby se každý snímek analyzoval do určitého časového okamžiku, a to přibližně 33 ms.

Protože nejdelší dobu pro výpočet vyžaduje samotná detekce, a to jakákoliv její metoda, byl čas nejdůležitějším kritériem pro volbu jejich nastavení. Mimo toto nastavení lze výpočet také urychlit změnou velikosti analyzovaného obrazu. S menším počtem pixelů se sníží i výpočetní čas. To s sebou ovšem nese i snížení schopnosti rozpoznávání. Proto je nutné určit hranici, kdy je toto snížení ještě přípustné a má minimální vliv.

10.1 Ověření metody hledání okrajů

V případě detekce hledáním okrajů se nastavení volilo na hranici, kdy už začínalo docházet k poklesu schopnosti detekce značek. Tím se snížil počet falešných detekcí a urychlil se tak výpočet. Toto nastavení odpovídalo hodnotě středu prahových hodnot „60“ a podílu rozptylu „0,2“. Při tomto nastavení se doba celkové analýzy jednoho snímku snížila až do potřebných 33 milisekund. To lze ještě navíc ovlivnit zmenšením snímku. V tomto případě to ale nebylo nutné pro zachování vyšší schopnosti rozpoznávání detekovaných oblastí.

Z výsledků aplikace na video vyplývá, že použití metody hledání okrajů funguje ve většině případů spolehlivě, což potvrdilo výsledky předešlého experimentu na jednotlivé snímky u vybraných značek. Ačkoliv se zde stále občas objevují falešně pozitivní detekce, i přes jejich eliminaci různými postupy popsanými výše, viz poměr stran, rozpoznávání tvarů atd., jsou tyto oblasti ve finále eliminovány prahovou hodnotou korelační shody, která tyto oblasti vyřadí jako nerozpoznatelné. Ve videozáznamu jsou vizualizovány červenou barvou, zatímco rozpoznané značky žlutou [22].

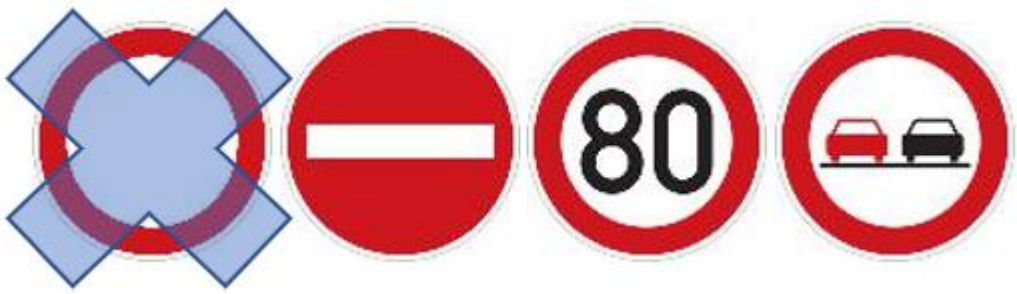
Nelze přímo určit procento úspěšnosti rozpoznávání všech značek, protože se jedná jen o pár minut záznamu, který slouží pouze pro ověření předešlého experimentu. Lze tedy teď jen usuzovat, které konkrétní značky nebo jejich typy jsou snáze rozpoznatelné.

Docházelo zde k několika falešně pozitivních detekcí, které se musely eliminovat. Veškeré nastavení se tak intuitivně dolaďovalo na základě toho, k jakým chybám docházelo. V první řadě se to týkalo prahových hodnot korelační shody. Ty se určily na základě detekovaných oblastí a procent shody, podle kterého zde byla určena daná značka. Tato hodnota se pak zvyšovala tak, aby se eliminovaly falešně pozitivní detekce, ale zároveň tak, aby bylo stále možné rozpoznat danou značku.

Ve výsledku byly tyto prahové hodnoty pro všechny základní tvary nastaveny na 92 %. V případě univerzálních tvarů (viz rozpoznávání tvarů) byla tato hodnota snížena, ačkoliv jejich shoda se šablonou dosahovala ve většině případů vysokého procenta. Z toho důvodu byla snížena pouze o část, a to na 80 %.

V několika případech docházelo k tomu, že byla daná značka určena špatně. Pokud se tato chyba objevovala vícekrát, byla šablona, která to způsobovala, odebrána z celkové množiny rozpoznatelných značek. Nelze je tak touto metodou dobře určit a byly vyloučeny jako nerozpoznatelné. Tímto postupem se sice snížilo množství detekce schopných značek, ale naopak se zvýšila schopnost přesného určení zbylých značek.

Jednalo se především o vzájemně podobné značky, jako je například „Zákaz vjezdu v obou směrech" (Obr. 10.1 – první), která je svým tvarem a barevným uspořádáním podobná většině dopravních značek, jen neobsahuje žádnou informaci uvnitř. Systém ji paradoxně zaměňoval i za značku „Zákaz vjezdu v jednom směru" (Obr. 10.1 – druhá), i když jsou už více vzájemně odlišné.



Obr. 10.1 Porovnání podobných dopravních značek

Další značkou, která byla vyloučena, je „Konec všech zákazů". Ačkoliv se příliš nepodobá žádné z ostatních značek, její korelační shoda byla v několika případech největší. Podle analýzy videozáznamu lze navíc usoudit, že právě tato značka způsobovala největší množství chyb, protože se shodovala jak s jinými značkami, tak s oblastmi, kde se ani žádná značka nenacházela. Je to způsobené nejspíše jejím jednoduchým vzhledem, bílá kruhová plocha přeškrtnutá tenkými černými pruhy, což se v případě korelace může jevit jako shoda s bílou plochou a černým vyobrazením informace na jiným dopravních značkách.

Dále pak značka „Zákaz zastavení" a „Zákaz stání" si jsou natolik podobné, že nemělo smysl se jejich odlišováním zabývat, protože mají i podobný význam. Považovaly se tak za stejné a zůstaly zachovány obě.

Naopak dopravní značka „Hlavní pozemní komunikace“ byla správně detekována a rozpoznána téměř vždy, a to bez ohledu na vliv okolností.

Toto testování tedy ukázalo, že nelze předem predikovat, které značky nebude možné rozpoznávat, ale že je u nich nejdříve nutné provést experiment.

10.2 Ověření metody prahování

V rámci porovnání výsledků všech použitelných metod se v druhém případě ověřovala metoda prahování, i když se nejevila podle výsledků experimentu jako vhodná pro tuto aplikaci, protože pro různé denní situace vyžaduje také různá nastavení prahových hodnot. Z toho důvodu se odzkoušela všechna možná nastavení této metody, a to jak pro konkrétní denní situaci, tak univerzální pro všechny. Cílem bylo především ověřit výsledky předešlého experimentu a také to, zda má smysl se s ní dále zabývat.

Dále pak bylo nutné zjistit, zda a za jakých okolností je vhodná k real-time detekci. K tomu bylo tedy potřeba změnit nastavení tak, aby neklesla schopnost detekce nebo samotného rozpoznávání, ale zároveň udržet výpočetní čas přibližně do daných 33 milisekund. Jako v předchozím případě se tím ověřila její celková funkčnost i to, zda je skutečně nutné toto nastavení měnit v závislosti na počasí a intenzitě osvětlení.

Oproti metodě s použitím hledání okrajů je využití metody prahování výpočetně více náročné, jak se ukázalo, a proto pro udržení nízkého výpočetního času se musí pracovat se snímky s nízkým rozlišením. V tomto případě se jednalo o rozlišení až 540×360 px. To samozřejmě ale vede k výslednému snížení schopnosti detekce a následnému rozpoznávání dané oblasti. Metoda vykazovala v mnoha případech časté falešně pozitivní detekce, které byly následně vlivem sníženého rozlišení snímku špatně rozpoznatelné, a proto často docházelo ke shodě s jinými šablonami, než o jakou se ve skutečnosti jednalo, nebo byly rozpoznány oblasti s určitou shodou, ve kterých se žádná značka nenacházela.

Tyto výsledky vedly k odzkoušení metody adaptivního thresholdingu, která byla prozatím pouze zmíněná. Jedná se o rozšíření této standardní metody. V původním experimentu se s ní nepočítalo, protože se její nastavení příliš nemění oproti standardnímu thresholdingu (prahování), a proto nebylo nutné se s ní zabývat jako s další metodou. Její výhodou je adaptace (přizpůsobení se) k různé intenzitě denního osvětlení. Kvůli tomu je ovšem navíc o něco více výpočetně náročná, a proto je nutné ji opět nastavit pro snížení doby výpočtu. Účelem teď tedy bylo zjistit, zda by byla vůbec schopná real-time detekce a zda by dosahovala větší úspěšnosti.

Výsledkem tohoto experimentu bylo, že se doba výpočtu mnohonásobně zvýšila navzdory několikanásobnému snížení rozlišení snímku, a to až na 1/9 původní testovací velikosti, tedy 540x360px. To již vylučuje možnost real-time detekce. Při tak malém rozlišení je navíc daleko menší pravděpodobnost přesného určování detekovaných oblastí.

Při celkovém shrnutí tak není metoda prahování, at už v jakékoliv její podobě, příliš vhodná k použití v real-time systémech pro detekci dopravních značek. Není nutné ji však úplně zavrhnout, ovšem v tomto případě je zde vhodnější metoda detekce, a to hledáním

okrajů, která vykazuje mnohonásobně lepší výsledky jak z hlediska výpočetního času, tak z hlediska spolehlivosti.

10.3 Ověření metody separování barev

Metoda separováním barev již podle poznatků z rešerše není příliš vhodná. To potvrdilo i experimentální testování, a proto je zde pouze zmíněná a její otestování proběhlo, jen aby byly zahrnuté všechny metody.

Výsledkem tedy bylo, že tato metoda skutečně není vhodná k detekci dopravních značek, ovšem i když byl každý snímek analyzován hned třikrát kvůli detekci všech barev, výpočetní čas byl do 30 ms. To bylo způsobené nejspíše tím, že při této detekci nevzniká tolik falešně pozitivních detekcí jako u jiných metod, a proto není nutné analyzovat příliš mnoho oblastí.

Jako u metody hledáním okrajů byla nejlépe v tomto případě detekována dopravní značka „Hlavní pozemní komunikace“. To je ovšem vše, ostatní značky jiných barev nebylo možné detekovat. Na to měla vliv pravděpodobně její bílá barva, protože se jí v obraze objevuje nejvíce (viz metoda separováním barev), a tak se neztrácí oproti jiným barvám.

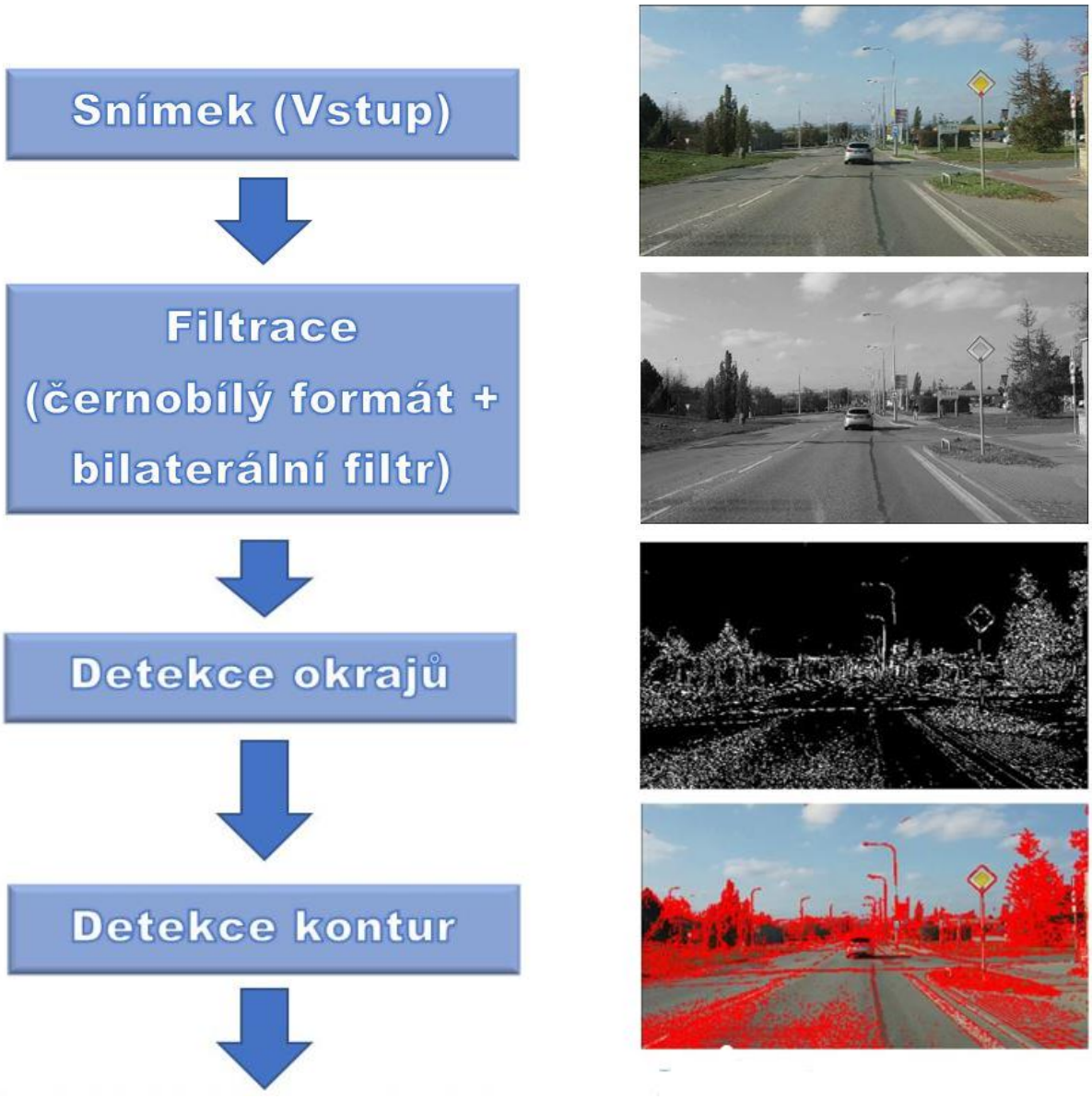
11. Koncepční návrh

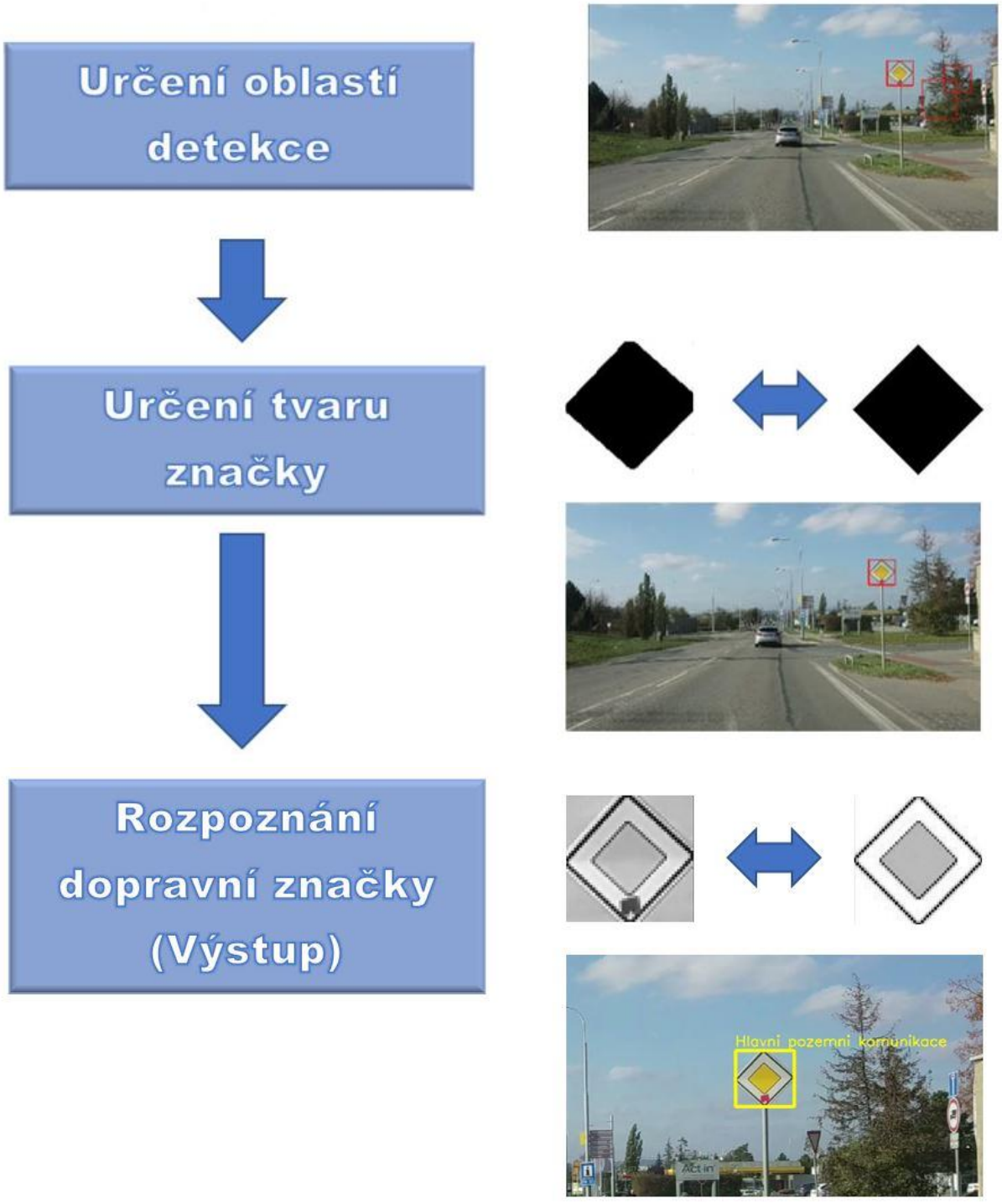
Koncepční návrh vychází z výsledků experimentu. Počítá se zde už jen s metodou detekce hledáním okrajů jako nejvhodnějšího řešení. Další postupy jako například rozpoznávání tvarů zůstávají stejné. Pro samotné rozpoznávání detekované oblasti se použije metoda korelace. Ačkoliv knihovna OpenCV nabízí několik typů této metody a jejich výsledky se ve většině případů příliš nelišily, byla zvolena primárně metoda „CCORR", protože celkově dosahovala lepších výsledků vůči ostatním. Ze všech těchto vybraných postupů se následně vytvořila jednoduchá API pro snadné používání.

11.1 Vytvoření API

API má sloužit uživateli pro detekci rozpoznávání dopravních značek v obraze, bez potřeby znalostí dané problematiky. Vstupem by měl být pouze snímek a výstupem pak rozpoznané značky a jejich pozice.

Vnitřní struktura této API je odvozena od struktury použité v experimentech (Obr. 11.1).





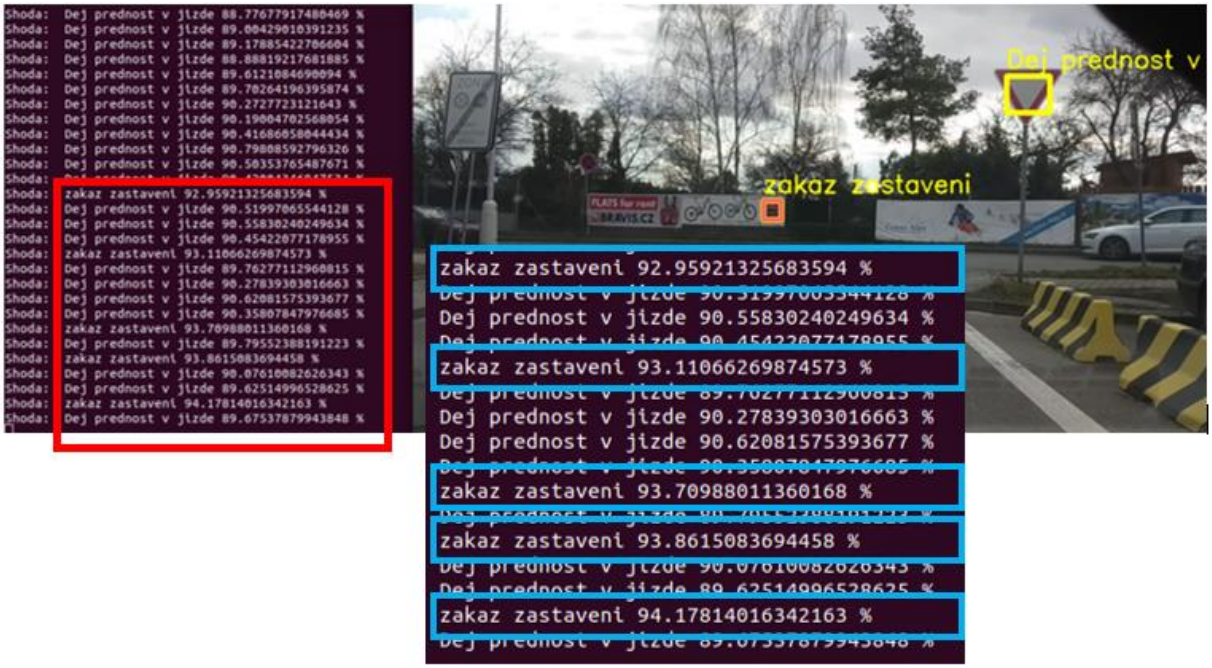
Obr. 11.1 Struktura API

Po vložení snímku se tento snímek převede na černobílý formát a vyfiltruje se pomocí bilaterálního filtru pro zachování hran, ale rozostření zbytku obrazu. Díky tomu lze pak snáze a přesněji detekovat okraje na základě gradientu intenzity, který využívá knihovna OpenCV. Na základě těchto okrajů se následně určí kontury určující oblasti detekce. Pro usnadnění výpočtu se část těchto kontur eliminuje, a to převážně ty, co jsou vzniklé zbylým šumem, a jsou tedy příliš malé na to, aby obsahovaly dopravní značku. Ohraničením zbylých kontur se určí oblasti detekce, které se následně vytřídí na základě

rozpoznaných tvarů těchto kontur. Tím se eliminují falešně pozitivní detekce a ostatní kontury se roztřídí podle jejich tvaru. V této fázi se tedy určily pozice jednotlivých značek a lze přistoupit k poslednímu kroku, a to samotnému rozpoznávání jednotlivých oblastí pomocí korelační metody typu „CCORR" v knihovně OpenCV. Výsledkem je pak seznam rozpoznaných značek a jejich pozice v obraze.

11.2 Návrh algoritmu pro eliminaci falešných pozitiv

Protože při aplikaci na videozáznam, kde se analyzuje sekvenčně několik snímků za sekundu, dochází občas v určitých oblastech ke ztrátě již rozpoznané značky, například z důvodu jiného osvětlení. Dále pak občas dojde ještě k falešnému rozpoznání. To má za následek, že se mezi správně detekované značky vměšují tyto nechtěné detekce, které je nutné eliminovat (Obr. 11.2).



Obr. 11.2 Příklad falešně rozpoznaných značek v sekvenci několika snímků

To lze vyřešit neustálým ověřováním všech detekovaných značek, zda jsou detekovány pozitivně, nebo negativně. Jak je vidět na průběhu analýzy videozáznamu, tak v sekvenci několika snímků a při správně detekované značce je tato detekce potvrzena v dalších snímcích (Obr. 11.3). V případě falešně pozitivních detekcí dojde k této detekci pouze zřídka, navíc zde ve většině případů bývá určena pokaždé jiná značka.

Obr. 11.3 Pozitivně rozpoznaná značka v sekvenci několika snímků

Díky tomu lze využít této paměti a nejdříve si každou detekovanou značku ověřit. K tomu byl vymyšlen algoritmus, který se stará o to, aby se těmto situacím předcházelo.

Ve zkratce lze říct, že každá značka musí být detekována vícekrát za sebou, než bude označena spolehlivě za rozpoznanou, v opačném případě je ignorována. Pokud už však není ověřená značka několikrát za sebou detekována, je opět následně odebrána. Protože ale občas dochází k nechtěným výpadkům v detekci, je nutné s tímto počítat, aby se nevyřazovaly správně rozpoznané značky.

Algoritmus tak pracoval na principu, který lze přirovnat k „prázdným nádobám“, jež se podle detekcí plnily, nebo naopak vylévaly. Každá detekovaná značka dostala přidělenou vlastní nádobu, která se vždy zčásti naplnila při její první detekci. Každá další detekce ji pak po menších částech dolévala. Pokud v některém dalším snímku detekovaná nebyla, byla tato nádoba zase zčásti vylitá. Důvod, proč se na počátku nádoba plní z větší části, spočívá v tom, aby v případě, že právě při další iteraci detekována nebude, nedošlo k její okamžité ztrátě. V této oblasti byly detekované značky pouze v paměti, ale ještě nebyly určené jako spolehlivě rozpoznané. Až při dosažení určité prahové hodnoty dojde k určení dané značky a výsledek je vypsán. Tímto se stává tato značka „imunní“ a k jejímu vyřazení může dojít až po vylití celé její nádoby. Protože tímto způsobem může být každá nádoba plněna donekonečna, například pokud vozidlo dlouho stojí před danou značkou, tak i k jejímu vylití by pak nemuselo nikdy dojít. Tomu se předchází saturací, kdy má každá nádoba omezený objem, přes který už nejde dále plnit. Ve výsledku se tak tedy eliminují falešně pozitivní detekce se zachováním ostatních značek.

12. Závěr

Hlavním cílem práce bylo na základě metod zpracování obrazu navrhnout vhodnou aplikaci pro rozpoznávání dopravních značek. K tomu byla využita knihovna OpenCV v programovacím jazyce Python, která umožňuje uživateli snazší práci s obrazem.

Nejdříve se provedla rešerše o běžně používaných metodách, ze které se vycházelo při následném návrhu experimentu, kde se tyto metody testovaly. Na to navazoval další cíl, a to nasbírání testovacích vzorků. Ty se sbíraly z videozáznamu pořízeného během jízdy vozidlem. Následně bylo vybráno ze všech pořízených snímků celkem pět dopravních značek, které se vyskytovaly nejčastěji nebo jejichž význam byl posouzen jako důležitější. Tím se vytvořila testovací množina dat, kde se každá značka vyskytovala v deseti různých scénách ve třech různých denních situacích (slunečno, zataženo, noc).

V dalším kroku se na základě této množiny ověřovaly různé metody, a to nejdříve pro detekci i s rozpoznáváním pomocí „cross-correlation" metody, kdy se prochází celý obraz se šablonou hledané značky a hledá se bod s nejvyšší shodou. Následně se použily různé metody detekce a rozpoznávání separovaně, což vedlo k urychlení a zpřesnění výsledků. Detekce se prováděla hledáním okrajů, prahováním obrazu a separováním barev. Každá tato metoda je již z určité části předprogramovaná v používané knihovně OpenCV, a tak se testovalo jejich různé nastavení pro dosažení nejlepších výsledků. Zbylé postupy a úpravy obrazu byly doprogramovány už jen podle potřeby na danou aplikaci. Pro určení detekovaných oblastí se ověřilo několik metod, ale pouze metoda korelace dosahovala přesných výsledků. Proto byla testována jako jediná k tomuto účelu. Její vysoká spolehlivost vedla i k aplikaci pro rozpoznávání detekovaných tvarů, zda se shodují s tvary dopravních značek. Díky tomu se dosáhlo lepších výsledků a eliminace falešně pozitivních detekcí. Původně se zde testovala pouze metoda aproximace, kde se vycházelo z počtu stran. To ji však omezovalo jen na přesné tvary, a proto často docházelo ke špatnému určení.

Z výsledků testovaných metod se určily ty, které bylo možné použít v real-time systému tak, že byly schopné analyzovat až 30 snímků za sekundu. Jejich nastavení se optimalizovalo na přijatelný poměr, aby dosahovaly co nejvyšší schopnosti detekce a zároveň nepřesáhly maximální čas pro výpočet. Tím vzniklo několik postupů pro detekci a rozpoznávání dopravních značek, které bylo nutné opět otestovat na nových vzorcích pro ověření jejich funkčnosti. V tomto bodě však už vstupem nebyly pouze jednotlivé snímky, ale nově pořízený videozáznam z jízdy vozidlem. Probíhalo tedy takzvané offline testování, kdy se ukázalo, jak se tyto navržené postupy chovají v běžném použití a zda se jejich výsledky neliší. Obecně lze říct, že všechny použité postupy se shodovaly s jejich předešlými výsledky, tím se zároveň potvrdilo i předešlé testování. Nicméně v konečném výsledku se jevil pouze jeden postup jako použitelný, a to kombinace metody hledáním okrajů a následné korelace, protože jediný ze všech zvládal i s jistou časovou rezervou dosáhnout nejlepších výsledků. Proto sloužil i jako vzor pro vytvoření finální API. Protože se ale v tomto novém testování objevují již všechny značky, docházelo občas k negativnímu určení. Pokud k tomu docházelo ve více případech, musela být tato značka odebrána. Z toho důvodu nelze prohlásit, že je tento způsob možné univerzálně použít na

jakoukoliv dopravní značku pouze s použitím šablony. Je proto nejspíše nutné uvažovat i metody strojového učení, které by tyto ztráty kompenzovaly.

Sledováním průběhů při detekci dopravních značek se nad rámec práce vytvořil optimalizační algoritmus, který dokázal eliminovat zbylé falešně pozitivní detekce, které narušovaly spolehlivost systému, a zároveň s vytvořenou pamětí kompenzoval občasné výpadky v detekci dané značky.

Při tomto nastavení je tedy systém schopen pracovat přibližně s 80% úspěšností. Ta se ovšem může měnit v závislosti na počtu a typu dopravních značek, které jsou vyžadovány k detekci. Se zvyšujícím se počtem nebo vzájemným podobným vzhledem rozpoznávací schopnost klesá. V opačném případě lze dosáhnout lepších výsledků, nicméně pro dosažení téměř stoprocentní spolehlivosti by bylo nutné navíc kombinovat tyto postupy například s metodami strojového učení.

Dále by se do budoucna mohl navrhnout vhodný hardware. Ten by měl dosahovat dostatečně vysokého výkonu, aby bylo možné v co nejkratším čase analyzovat jednotlivé snímky. Na to navazuje výběr kamery pro vytváření ostrých snímků v požadovaném rozlišení alespoň 960×540 px. Tyhle dvě věci je nutné mezi sebou optimalizovat, protože vyšší rozlišení s sebou nese vyšší nároky na výpočetní výkon.

POUŽITÁ LITERATURA

[1] LORSAKUL, Auranuch a Jackrit SUTHAKORN. Traffic Sign Recognition Using Neural Network on OpenCV: Toward Intelligent Vehicle/Driver Assistance Systém. *BART LAB*. [online]. [cit. 2020-06-14]. Dostupné z: http://bartlab.org/Dr.%20Jackrit's%20Papers/ney/1.TRAFFIC_SIGN_Lorsakul_ISR.pdf

[2] FORSON, Eddie. Recognising Traffic Signs With 98% Accuracy Using Deep Learning. *Towards data science.* [online]. 2017-08-24 [cit. 2020-06-14]. Dostupné z: https://towardsdatascience.com/recognizing-traffic-signs-with-over-98-accuracy-using-deep-learning-86737aedc2ab

[3] Detekce vlastních objektů v obrázku pomocí Haar Cascade. *ITnetwork.cz* [online]. 2020 [cit. 2020-06-14]. Dostupné z: https://www.itnetwork.cz/python/video/detekce-vlastnich-objektu-v-obrazku-pomoci-haar-cascade

[4] BHATT, Samir, Bhaskar TRIVEDI, Ankur DEVANI a Hemang BHIMANI. Understanding the Technology behind Traffic Sign Recognition (TSR) Systems. *Design and Reuse* [online]. 2020 [cit. 2020-06-14]. Dostupné z: https://www.design-reuse.com/articles/41154/traffic-sign-recognition-tsr-system.html

[5] LAGUNA, Rubén, Rubén BARRIENTOS, L. Felipe BLÁZQUES a Luis J. MIGUEL. Traffic sign recognition application based on image processing techniques. *Science Direct* [online]. 2014, s. 104-109 [cit. 2020-06-14]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1474667016416009

[6] Porovnávání vzorů a detekce rohů a čar v obrázcích v Pythonu. *ITnetwork.cz* [online]. 2020 [cit. 2020-06-14]. Dostupné z: https://www.itnetwork.cz/python/video/porovnavani-vzoru-a-detekce-rohu-a-car-v-obrazcich-v-pythonu

[7] DEWAN, Prachi, Rekha VIG a Bival Kumar DAS. An Overview of Traffic Signs Recognition Methods. *International Journal of Computer Applications* [online]. 2017, (11), s. 7–10 [cit. 2020-06-14]. DOI: 10.5120/ijca2017914524. Dostupné z: https://pdfs.semanticscholar.org/5fd7/eca5432e4e4038dfc2ad425fa5f3e8739f0b.pdf

[8] Počítačová grafika. *Aldebaran* [online]. [cit. 2020-06-14]. Dostupné z: https://www.aldebaran.cz/onlineskola/etapy/grafika/barvy.htm

[9] HORDĚJČUK, Vojtěch. Genetický algoritmus. *Voho* [online]. 2008-2020 [cit. 2020-06-14]. Dostupné z: https://www.aldebaran.cz/onlineskola/etapy/grafika/barvy.htm

[10] GANDHI, Rohith. Support Vector Machine — Introduction to Machine Learning Algorithms. *Towards data science* [online]. 7.6.2018 [cit. 2020-06-14]. Dostupné z: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[11] Detekce objektu v obrázku podle barvy a detekce okrajů: Sobel. *Image Processing Fundamentals* [online]. 2020 [cit. 2020-06-15]. Dostupné z: https://www.itnetwork.cz/python/video/python-detekce-objektu-v-obrazku-podle-barvy-a-detekce-okraju

[12] Detekce objektu v obrázku podle barvy a detekce okrajů. *ITnetwork.cz* [online]. 2020 [cit. 2020-06-14]. Dostupné z: https://www.itnetwork.cz/python/video/python-detekce-objektu-v-obrazku-podle-barvy-a-detekce-okraju

[13] ORBAN, Cosimo. Detecting circular shapes using contours. *Authentise* [online]. 19.4.2016 [cit. 2020-06-14]. Dostupné z: https://www.authentise.com/post/detecting-circular-shapes-using-contours

[14] CANU, Sergio. Simple shape detection. *PySource* [online]. 25.9.2018 [cit. 2020-06-15]. Dostupné z: https://pysource.com/2018/09/25/simple-shape-detection-opencv-with-python-3/

[15] Thresholding a analýza obrázků v Pythonu: AdaptiveThreshold. *ITnetwork.cz* [online]. 2020 [cit. 2020-06-15]. Dostupné z: https://www.itnetwork.cz/python/video/thresholding-a-analyza-obrazku-v-pythonu

[16] Seznam dopravních značek v Česku. *Wikipedie* [online]. 31.5.2020 [cit. 2020-06-15]. Dostupné z: https://cs.wikipedia.org/wiki/Seznam_dopravn%C3%ADch_zna%C4%8Dek_v_%C4%8Cesku

[17] Template Matching. *OpenCV documentation* [online]. 31.12.2019 [cit. 2020-06-15]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

[18] Canny Edge Detection. *OpenCV* [online]. [cit. 2020-06-15]. Dostupné z: https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html

[19] Structural Analysis and Shape Descriptors: findContours. *OpenCV documentation* [online]. 31.12.2019 [cit. 2020-06-15]. Dostupné z: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours

[20] Basic Thresholding Operations. *OpenCV documentation* [online]. 31.12.2019 [cit. 2020-06-15]. Dostupné z: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours

[21] Segmentation: Thresholding. *Image Processing Fundamentals* [online]. [cit. 2020-06-15]. Dostupné z: http://www.mif.vu.lt/atpazinimas/dip/FIP/fip-Segmenta.html#Heading118

[22] Rozpoznávání dopravních značek. In: *Youtube* [online] 24.6.2020 [cit. 2020-06-24]. Dostupné z: https://youtu.be/jlT3uK3hzvA. Kanál uživatele Josef Šmíd.

SEZNAM OBRÁZKŮ

Obr. 3.1 Mapy korelačních hodnot detekovaných značek 13
Obr. 3.2 „Podpisy" jednotlivých tvarů 15
Obr. 5.1 Testovací vzorky dopravních značek 17
Obr. 6.1 Detekované vzájemné body metodou BFMatch 18
Obr. 6.2 Vzorový obrázek 19
Obr. 6.3 Mapa korelačních hodnot (vlevo) a detekovaná značka (vpravo) 19
Obr. 6.4 Porovnání korelačních metod – Hlavní pozemní komunikace 20
Obr. 6.5 Porovnání korelačních metod – Dej přednost v jízdě 20
Obr. 7.1 Příklad horizontálního a vertikálního Sobelova filtru o velikosti 3×3 [11] 22
Obr. 7.2 Testovaný obrázek bez filtrace (vlevo) a s bilaterálním filtrem (vpravo) 23
Obr. 7.3 Detekce okrajů bez filtrace (vlevo), s bilaterálním filtrem (vpravo) 23
Obr. 7.4 Detekce okrajů bez filtrace (vlevo), s bilaterálním filtrem (vpravo) 24
Obr. 7.5 Detekce okrajů bez filtrace (vlevo), s bilaterálním filtrem (vpravo) 24
Obr. 7.6 Detekované kontury (vlevo) a oblasti detekce (vpravo) bez filtru 25
Obr. 7.7 Detekované kontury (vlevo) a oblasti detekce (vpravo) s bilatrálním filtrem 25
Obr. 7.8 Průběh detekcí hledáním okrajů pomocí Gaussova filtru 26
Obr. 7.9 Průběh detekcí hledáním okrajů pomocí bilaterálního filtru 26
Obr. 7.10 Průběh detekcí hledáním okrajů pro slunečno pomocí Gaussova filtru 27
Obr. 7.11 Průběh detekcí hledáním okrajů pro slunečno pomocí bilaterálního filtru 28
Obr. 7.12 Průběh detekcí hledáním okrajů pro oblačno pomocí Gaussova filtru 28
Obr. 7.13 Průběh detekcí hledáním okrajů pro oblačno pomocí bilaterálního filtru 29
Obr. 7.14 Průběh detekcí hledáním okrajů pro noc pomocí Gaussova filtru 29
Obr. 7.15 Průběh detekcí hledáním okrajů pro noc pomocí bilaterálního filtru 30
Obr. 7.16 Průběh detekcí hledáním okrajů pro jednotlivé značky – Slunce 31
Obr. 7.17 Průběh detekcí hledáním okrajů pro jednotlivé značky – Zataženo 31
Obr. 7.18 Průběh detekcí hledáním okrajů pro jednotlivé značky – Noc 32
Obr. 7.19 Adaptivní thresholding, použití Gaussova rozložení 33
Obr. 7.20 Adaptivní thresholding, použití průměrné hodnoty 33
Obr. 7.21 Originální obrázek (vlevo), černobílá verze (vpravo) 34
Obr. 7.22 Metoda cv.THRESH_OTSU (vlevo) a metoda cv.THRESH_TRIANGLE (vpravo) 35
Obr. 7.23 Metody základní (vlevo), metody inverzní (vpravo) 35
Obr. 7.24 Detekované kontury (vlevo) a oblasti detekce (vpravo) 36
Obr. 7.25 Průběhy detekcí všech značek ve všech denních situacích 36
Obr. 7.26 Průběhy detekcí všech značek – Slunce 37
Obr. 7.27 Průběhy detekcí všech značek – Zataženo 37
Obr. 7.28 Průběhy detekcí všech značek – Noc 38
Obr. 7.29 Graf průběhů detekcí jednotlivých značek – Slunce 39
Obr. 7.30 Graf průběhů detekcí jednotlivých značek - Zataženo 39
Obr. 7.31 Graf průběhů detekcí jednotlivých značek – Noc 40
Obr. 7.32 Separování červené barvy 42
Obr. 7.33 Separování modré barvy 42
Obr. 7.34 Separování bílé barvy 42
Obr. 7.35 Metoda detekce separováním barev – Slunce 43

Obr. 7.36 Metoda detekce separováním barev – Zataženo 43
Obr. 7.37 Metoda detekce separováním barev – Noc 44
Obr. 7.38 Metoda detekce separováním barev – Všechny denní situace 45
Obr. 8.1 Šablony tvarů dopravních značek pro korelaci 46
Obr. 8.2 Detekované kontury (vlevo), oblasti detekce (vpravo) 47
Obr. 8.3 Tvary některých detekovaných kontur 47
Obr. 8.4 Oblasti detekce po eliminaci falešných detekcí 48
Obr. 9.1 Detekované a rozpoznané značky v obraze 50
Obr. 9.2 Porovnání korelačních metod pro rozpoznávání 51
Obr. 10.1 Porovnání podobných dopravních značek 54
Obr. 11.1 Struktura API 58
Obr. 11.2 Příklad falešně rozpoznaných značek v sekvenci několika snímků 59
Obr. 11.3 Pozitivně rozpoznaná značka v sekvenci několika snímků 60

SEZNAM TABULEK

Tabulka 7.1 Hodnoty barevného formátu HSV a jejich zvolený rozptyl 41
Tabulka 9.1 Porovnání metod detekce 50

SEZNAM ZKRATEK

API	Application Programming Interface
OpenCV	Open Computer Vision
SVM	Support Vector Machine
HSV	Hue, Saturation, Value
BGR	Blue, Green, Red

ELEKTRONICKÉ PŘÍLOHY

Excel – Grafy výsledků všech použitých metod pro detekci a rozpoznávání

Testovací snímky dopravních značek

Skripty (Python) s přílohami:

Metody detekce:

- Metoda hledáním okrajů (edge_based_detection.py)
- Metoda separováním barev (color_based_detection.py)
- Metoda prahování obrázku (thresh_based_detection.py)
- Algoritmus hledání HSV hodnot (HSV_callback_values.py)

Metody rozpoznávání s detekcí:

- Metoda MatchTemplate (matchTemplate_recognition.py)
- Metoda MatchTemplate v černobílém formátu (matchTemplate_recognition_gray.py)
- Metoda BFMatcher (sign_recognition_ORB.py)

Rozpoznávání – hlavní program:

- Šablony tvarů
- Šablony dopravních značek
- Porovnání korelačních metod (CM_color_images.py)
- Porovnání korelačních metod v černobílém formátu (CM_gray_images.py)
- Rozpoznávání API (Recognition_API.py)
- Offline testování videa (Video_Recognition_Standard.py)
- Offline testování videa s optimalizačním algoritmem (Video_Recognition_Filter_Algorithm.py)